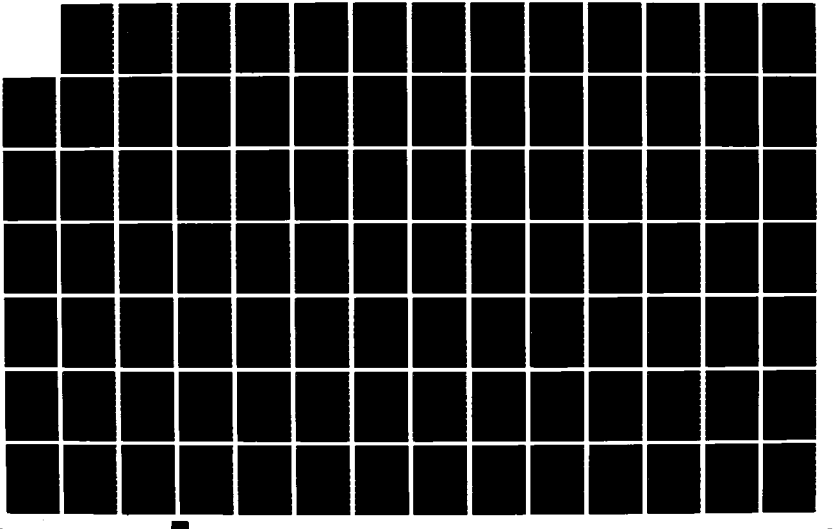
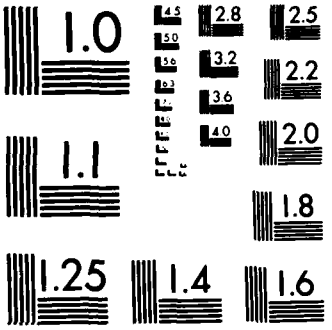
AD-A171 842 ALGORITHMS AND SOFTWARE FOR SOLVING COUPLED DISCRETE-TIME RICCATI EQUATIO..(U) ILLINOIS UNIV AT URBANA DECISION AND CONTROL LAB P J WEST AUG 86 DC-88 1/2

UNCLASSIFIED N00014-84-C-0149 F/G 12/1 NL





August 1986

UILU-ENG-86-2233
DC-88

(12)

COORDINATED SCIENCE LABORATORY
College of Engineering
Decision and Control Laboratory

ALGORITHMS AND SOFTWARE FOR SOLVING COUPLED DISCRETE-TIME RICCATI EQUATIONS VIA THE L-A-S LANGUAGE

Phillip James West

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A171842

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release, distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) (DC-88) UILU-ENG-86-2233	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Laboratory, Univ. of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State and ZIP Code) 1101 W. Springfield Avenue Urbana, Illinois 61801	7b. ADDRESS (City, State and ZIP Code) 800 N. Quincy Street Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION JSEP 8b. OFFICE SYMBOL (If applicable) N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-84-C-0149
8c. ADDRESS (City, State and ZIP Code) 800 N. Quincy Street Arlington, VA 22217	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: N/A PROJECT NO.: N/A TASK NO.: N/A WORK UNIT NO.: N/A
11. TITLE (Include Security Classification) Algorithms and Software for Solving Coupled Discrete-Time	
12. PERSONAL AUTHOR(S) Phillip James West	
13a. TYPE OF REPORT Interim Technical, final 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) August 1986 15. PAGE COUNT 131
16. SUPPLEMENTARY NOTATION N/A	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Riccati equations, L-A-S language, CAD (computer-aided design)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis conducts an in-depth study of the computational issues associated with solving a set of coupled discrete-time Riccati equations. Briefly, the organization of this study is as follows. First, the problem is motivated by discussing two game situations which give rise to coupled discrete-time Riccati equations. Next, the computational aspects of solving these coupled equations are investigated. Finally, algorithms and software are produced that iterate these equations in a numerically robust and computationally efficient manner. The thesis carries the coupled Riccati problem from formulation to software implementation with several theoretical advances along the way. However, the major contribution of this work is the Riccati solution method - i.e., the algorithms and software which solve the problem. As the algorithms are formulated, structured, and subsequently coded, the software engineering factors that influence good software design are addressed. Furthermore, the coupled Riccati software developed here is integrated into a well-known Computer-Aided Design (CAD) software package. Thus, the informal computer user has easy access to software which solves both single and coupled Riccati equations.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code) 22c. OFFICE SYMBOL NONE

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE UNCLASSIFIED

ALGORITHMS AND SOFTWARE FOR SOLVING COUPLED DISCRETE-TIME RICCATI EQUATIONS VIA THE L-A-S LANGUAGE

BY

PHILLIP JAMES WEST

B.S., University of Illinois, 1980
M.S., University of Illinois, 1982

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1986

Thesis Advisor: Professor William R. Perkins

Urbana, Illinois

ALGORITHMS AND SOFTWARE FOR SOLVING COUPLED DISCRETE-TIME RICCATI EQUATIONS VIA THE L-A-S LANGUAGE

Phillip James West, Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1986

This thesis conducts an in-depth study of the computational issues associated with solving a set of coupled discrete-time Riccati equations. Briefly, the organization of this study is as follows. First, the problem is motivated by discussing two game situations which give rise to coupled discrete-time Riccati equations. Next, the computational aspects of solving these coupled equations are investigated. Finally, algorithms and software are produced that iterate these equations in a numerically robust and computationally efficient manner. The thesis carries the coupled Riccati problem from formulation to software implementation with several theoretical advances along the way. However, the major contribution of this work is the Riccati solution method - i.e., the algorithms and software which solve the problem. As the algorithms are formulated, structured, and subsequently coded, the software engineering factors that influence good software design are addressed. Furthermore, the coupled Riccati software developed here is integrated into a well-known Computer-Aided Design (CAD) software package. Thus, the informal computer user has easy access to software which solves both single and coupled Riccati equations.

ACKNOWLEDGEMENTS

There are so many people to thank for their assistance. Professor W. R. Perkins heads the list for his endless supply of wisdom and encouragement. Professors T. Basar, J. B. Cruz, Jr., H. V. Poor, and A. Sameh gave their time freely when it was needed. My folks provided the family environment and stability that nurtured the internal security needed to pursue this task. My friend Bruce Mather is to be thanked for his acute spiritual guidance. My sister Robin deserves special recognition for being my roommate for the last two years. She has been a continual source of inspiration during this time. Most of all I must express my sincerest love and respect to Lorri Perkins, soon to be my wife. Her unyielding love and support have given purpose and future to my life.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This thesis conducts an in-depth study of the computational issues associated with solving a set of coupled discrete-time Riccati equations. Briefly, the organization of this study is as follows. First, the problem is motivated by discussing two game situations which give rise to coupled discrete-time Riccati equations. Next, the computational aspects of solving these coupled equations are investigated. Finally, algorithms and software are produced that iterate these equations in a numerically robust and computationally efficient manner. The thesis carries the coupled Riccati problem from formulation to software implementation with several theoretical advances along the way. However, the major contribution of this work is the Riccati solution method - i.e., the algorithms and software which solve the problem. As the algorithms are formulated, structured, and subsequently coded, the software engineering factors that influence good software design are addressed. Furthermore, the coupled Riccati software developed here is integrated into a well-known Computer-Aided Design (CAD) software package. Thus, the informal computer user has easy access to software which solves both single and coupled Riccati equations.

1.1 Motivation

This section motivates the computational study of coupled discrete-time Riccati equations. Historical background is presented to give the proper setting. A short discussion of the scope of the thesis contribution follows to provide some breadth to the findings. But the set of computational issues is the key focus of this work and the software design and implementation are of fundamental concern. Hence, the Computer-Aided Control System Design (CACSD) field is introduced. Then, the L-A-S CACSD language is reviewed.

1.1.1 Background

The theory of optimal control has reached a significant level of maturity as evidenced by the number of textbooks written on the subject [e.g., 1,2,3]. A classic problem often discussed in introductory courses is the Linear Quadratic (LQ) regulator using state feedback. Here, one finds that the optimal control is obtained by solving a single Riccati equation. Originally, the solution to the dual (filtering) problem was given by Kalman [4]. Subsequently, both continuous-time [1,2,3] and discrete-time [5,6] versions of the Riccati equation have been studied in detail. A key feature of optimal control problems is that there is a single control agent or Decision Maker (DM).

A more interesting situation occurs when there are two or more DMs controlling the underlying dynamic system. A straightforward generalization of the state-feedback regulator problem to multiple DMs, each with its own LQ objective functional, leads to the feedback Nash equilibrium concept [7]. Here, one finds that the feedback Nash equilibrium solution, if it exists, is obtained by solving a set of *coupled* Riccati equations. However, the numerical solution of these coupled Riccati equations is substantially more difficult to characterize. Furthermore, conditions insuring existence and uniqueness of fixed points of these coupled equations have not been produced. Nevertheless, several results concerning the general LQ continuous-time problem have appeared [7-12]. By contrast, the discrete-time case has received considerably less attention. Indeed, most of the results on discrete-time feedback Nash solutions may be found in [7] or [13]. But these authors point out the need for research into the computational aspects of solving these game problems.

In this thesis, we solve for the so-called linear, state-feedback (perfect-state information) Nash equilibrium of a discrete-time descriptor system. For simplicity only the two DM case is considered. The generalization to three or more DMs remains an open problem. The material presented here suggests an obvious approach towards extending the theory. The main contribution of this work, however, is the development of algorithms and software suitable for solving coupled discrete-time Riccati equations arising from LQ feedback Nash games. Additionally, solutions to

large-scale problems and single-player (control) problems follow immediately from the descriptor-variable formulation taken here. The convergence behavior of these equations is investigated which results in a contraction mapping argument guaranteeing existence and uniqueness of a solution to the infinite-horizon Nash game problem. Also, a new type of game theory called *multirates* is discovered via asymptotic analysis. The tools of asymptotic analysis are also used in producing the contraction mapping result. The descriptor-variable game formulation as well as multirate game theory have not been examined until now. These theoretical contributions enable studies of a significantly larger class of LQ Nash game problems. Moreover, there are several practical situations where this new theory yields more accurate and/or numerically appealing models.

Although theory is an integral part of this work, the end product is software. Hence, software engineering is a key issue. That is, the design and implementation of the Riccati software ought to comply with the standards and practices currently used for software development. This approach ultimately assures the quality of the final package. However, before embarking on a detailed discussion of the algorithms and the software structure, it is necessary to expand on this last idea more fully.

1.1.2 Computer—Aided Control System Design

A study such as this falls under the heading of Computer-Aided Control System Design (CACSD). Essentially, this research field strives to provide the control system and related communities with high-quality, reliable, numerically robust algorithms and software. CACSD is still a relatively young area of research (about 5 years old). This remark is supported by the observation that formal conferences on CACSD are relatively new [e.g., 14-16]. There are several problems, such as linear least squares or generalized eigenvalues-eigenvectors, that provide a basis for solving more complicated control, system, and estimation problems. It is extremely important that stable numerical methods are used for solving these simple problems. Then solutions to more

complicated problems rest on a firm algorithmic foundation. Some tools of modern numerical analysis which are finding significant utility in CACSD include orthogonal transformations [17], Householder transformations [17], Singular Value Decomposition (SVD) [17,18], invariant imbedding techniques [19], and descriptor variable formulations [19,20]. Some of these tools are useful for obtaining matrix forms with special structure (e.g., block triangular, Hessenberg, real Schur form) while others circumvent difficulties encountered in poorly or ill-conditioned problems. As a consequence of the CACSD approach, one can arrive at the solution to a problem in a straightforward and computationally efficient manner by exploiting the structure imposed by a particular technique. A recent and comprehensive survey of the preceding ideas may be found in [21].

With regards to robust numerical software, it is widely recognized [e.g., 21] that the EISPACK [22,23] and LINPACK [24] software packages are well-suited for generalized eigenvalue-eigenvector problems and linear equation problems, respectively. Both packages are coded in the FORTRAN [25] programming language. Furthermore, each package has demonstrated numerical superiority over the years. Hence, this software is a natural starting point for building the coupled Riccati equation algorithms. Sometimes a CACSD package consists of a collection of subroutines, often using EISPACK and/or LINPACK as the lowest level routines. RICPACK [26], a software package for single algebraic Riccati equations, is an example of a CACSD package with this structure. Other times a more substantial undertaking yields a package capable of solving a broad range of problems.

Despite the current evolutionary state of affairs, several trends are apparent. For instance, the large CACSD software packages emerging today may be classified according to the following groups: menu-driven, command-driven, expert systems, and languages. Although these categories are distinct, examples of packages can be found that possess elements of more than one group. Because of the volatility of CACSD packages, specific examples of each type are difficult to produce without dating this text. Nevertheless, a good overview of software packages available today may

be found in [27]. In particular, we mention that SIMNON [28] is representative of a command-driven package whereas L-A-S [27, pp. 243-261] qualifies as a CACSD language.

Each CACSD software group has its strengths and weaknesses in terms of time invested by the user. For example, menu-driven packages have the advantage that the infrequent user will probably spend a minimal amount of time (re)learning how to interact with the package by virtue of the menu-driven environment. However, the disadvantages include the fact that working through pages of menus is ultimately time-consuming. More importantly, if a solution procedure does not exist as one of the choices on the menus, then the problem is quite likely unsolvable by the given package. On the other extreme, CACSD languages typically require much more time (e.g., hours) to (re)gain familiarity with the package. But, once mastered, an almost limitless class of problems may be studied depending on the richness of the language.

Motivated by the desire to conduct systematic numerical studies of LQ Nash games in discrete-time as well as the need to efficiently manipulate matrices in a user-friendly environment, the decision is made to integrate the coupled Riccati software into the L-A-S language [27, pp. 243-261]. In order to explain the subtleties of the software implementation, a brief review of the L-A-S language package is required.

1.1.3 The L–A–S Language

BASIC, FORTRAN, and PASCAL are standard programming languages. Each possesses qualities and attributes that are characteristic of almost any ordinary programming language in use today (e.g., subroutine capabilities). It is desirable that a CACSD language parallels the organizational model set by these familiar and well-established computer languages. Furthermore, for control and linear system problems a sophisticated matrix environment is mandatory. In addition, frequency domain techniques require the analysis and manipulation of matrices of polynomials. It is according to these prerequisites that the L-A-S language was created.

L-A-S stands for Linear Algebra and Systems. Furthermore, L-A-S is a CACSD language in the strict sense of the word. That is, L-A-S conforms to standard computer science definitions for the syntactical specification of a programming language [see, 27, pp. 243-261, and 29]. In addition, L-A-S has been tested by industry and academia for over a decade at over a dozen locations around the world. Numerically speaking, L-A-S is based upon the EISPACK [23] and LINPACK [24] software. Also, the NCAR [30] graphics package is employed to provide 2D and 3D plotting capabilities. In summary, there is ample evidence [27,29,31-36] available to support the claim that L-A-S is a bona fide CACSD language.

In the normal interactive mode, the user types statements directly in the L-A-S language interpreter. Each statement is either a command to the interpreter (e.g., put L-A-S into program debug mode) or a request to perform some kind of calculation. The former instructs L-A-S to display or modify various status information concerning the current L-A-S work session. The latter invokes the L-A-S language parser which subsequently calls upon the FORTRAN subroutines needed to process the desired computation.

The fundamental concept behind any L-A-S statement is the L-A-S *operator*. Essentially, operators combine input data, perform some desired calculation, and generate output data. The utility of L-A-S operators as algorithmic "building blocks" has been established [27,35,36]. Thus, even though a single operator may not be available to solve a particular problem, it is quite likely that the desired result can be obtained by concatenating several "lower-level" operators. The L-A-S operators are divided into five groups: Input/Output, Data Handling, Linear Algebra, Control Systems, and L-A-S Program Control. Presently, there are more than 100 L-A-S operators. Also, the user may define up to 100 matrices with the total number of matrix elements not exceeding 50,000. The maximum order of any particular matrix is not explicitly limited.

L-A-S programs are written by combining one or more operator statements. Should questions arise, an extensive on-line help facility containing detailed information about L-A-S language usage is at the disposal of the user. In totality, L-A-S and its supporting software consist of over 20,000

lines of FORTRAN code (1977 Standard).

The L-A-S language will be used extensively throughout this dissertation. Therefore, it is assumed that the reader is adequately familiar with L-A-S so that the L-A-S programs presented here are easily understood.

1.2 Thesis Problem and Contribution

This section discusses the actual computational problem studied in this thesis and the specifics of the contribution of this work. Because the coupled discrete-time Riccati equations analyzed here are deeply rooted in Nash game theory, two game scenarios which lead to the solution of coupled discrete-time Riccati equations are developed. First, an exposition on descriptor-variable Nash games is presented. Then, multirate descriptor Nash games are introduced. The mathematical rigor associated with each problem is deferred until Chapter 2. The purpose of this discussion is to elucidate the theoretical novelty as well as the practical applicability of descriptor games. In particular, multirate games are extremely useful for formulating optimization problems involving digital communication channels operating at different rates.

Next, the details of the thesis contribution are highlighted. The computational obstacles pertaining to iterating two coupled discrete-time Riccati equations are delineated. The procedure by which they are overcome is outlined and justified. Relevant theoretical issues (such as convergence in the limit as the number of iterations tends toward infinity) and numerical issues (such as preserving symmetry) are addressed. Finally, the software implementation is described. Since the Riccati software is integrated into the L-A-S language, additional care must be taken to insure that the top-level Riccati routines conform to the L-A-S interfacing protocol. Also, the low-level routines must be engineered properly. Hence, structured programming and modularity concepts are discussed.

1.2.1 Introduction

In this subsection, we describe two distinct, yet related problems where coupled discrete-time Riccati equations arise. Both problem formulations represent new contributions to the theory of Nash games utilizing perfect state information. However the fact that both problems lead to the solution of two coupled Riccati equations is the main reason for their inclusion in this thesis. The ensuing discussion is primarily qualitative.

1.2.1.1 Descriptor Games

A linear shift-invariant (LSI) discrete-time, descriptor system takes the form :

$$E\ x(k+1) = A\ x(k) + B_1 u_1(k) + B_2 u_2(k) \qquad (k \geq 0,\ E\ x(0) = E\ x_0) \qquad (1.2.1)$$

$$y_1(k) = C_1 x(k) \qquad (1.2.2a)$$

$$y_2(k) = C_2 x(k)\ . \qquad (1.2.2b)$$

This game problem has two decision makers, DM1 and DM2. The discrete-time dynamic system is evolving at a rate indexed by the integer-valued variable, k. $x(k) \in \boldsymbol{R}^n$, $u_1(k) \in \boldsymbol{R}^{p_1}$, $u_2(k) \in \boldsymbol{R}^{p_2}$, $y_1(k) \in \boldsymbol{R}^{m_1}$, and $y_2(k) \in \boldsymbol{R}^{m_2}$. The matrix E is square and is assumed to be nonsingular to numerical precision. The standard state-space formulation is recovered by multiplying (1.2.1) by E^{-1}. The system being described is depicted in Figure 1.

Actually, the reasons for choosing a descriptor-variable system are more compelling than is first apparent. To begin with, many physical systems undergo a modelling phase, during which time the physical laws of nature are applied to the problem. Often, the result of this process is a static and/or dynamic (i.e., algebraic and/or difference equations) descriptor-variable description of the system plant. The matrix E plays the role of a mass matrix (when Newton's $F = MA$ is applied), or a sparse interconnection matrix (for distributed parameter systems), or a very singular matrix (for economic dynamic games). Whatever the case, it is not desirable, or even feasible, to go to the standard state-space description by inverting E. Second, the theoretical aspects of descriptor-variable systems are a relevant issue. Allowing E to be singular ultimately enlarges the

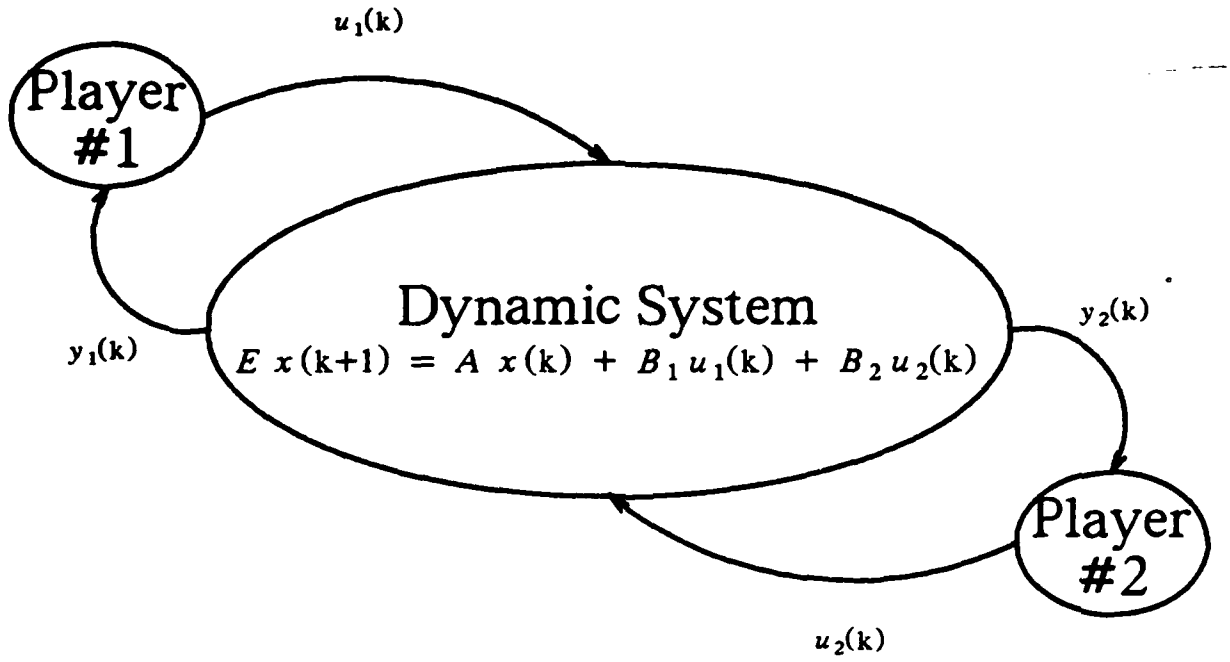


Figure 1. Two-Player, Discrete-Time, Descriptor Game.

class of problems that can be studied by game theorists. Last but not least, the descriptor-variable formulation is numerically superior to state-space ones for two reasons. Inverting E may be numerically impossible or inversion of E would destroy any inherent structure (e.g., sparsity) in that matrix. These facts and more support the philosophy that descriptor-variable formulations of optimal-control and dynamic-game problems are physically, theoretically, and numerically superior to state-space formulations.

Each DM has an associated LQ cost functional which is to be minimized. We solve for the linear, state-feedback (perfect-state information) Nash equilibrium solution which, by definition, obeys the principle of optimality. This involves extending the well-known single-player optimization result to descriptor games. This, in turn, leads to the solution of two coupled discrete-time Riccati equations.

1.2.1.2 Multirate Descriptor Games

This subsection introduces the theory of multirate descriptor games. The idea is that one DM is constrained to control the system at a rate which is slower than the other DM. It turns out that multirate descriptor games can be formulated and solved via single-rate descriptor game theory if appropriate limiting arguments are constructed. In addition, there are several practical situations where the optimization problem is more accurately modelled by multirates than by single rates.

Singular perturbation techniques are used successfully to exploit the presence of time scales within continuous-time [37] and discrete-time systems [38]. These applications deal with time scales that are inherent in the underlying system plant (i.e., the system's eigenvalues). By contrast, relatively little attention has been paid to the case where time scales are introduced by the control. Consider such a case in a multiple-decision, discrete-time setting where each DM is constrained to control the system at a different rate. Although an analysis tool has been developed for discrete-time systems with multirate samplers [6], it is not suitable for multiple-decision, optimization problems.

Multirate descriptor games begin with the LSI discrete-time, noncooperative game problem of the last subsection and constrain the DMs to play at different rates. As before, we determine the feedback (perfect state) Nash solution [7] which leads to the *periodic* solution of two coupled discrete-time Riccati equations. Again, we consider only two DMs. However, the results can be extended to multiple DM situations as well. In addition, we restrict attention to those rates which are related to all others by a positive *integer* constant. It is straightforward to generalize to rates which are related by *rational* constants. The fast player (DM1) plays at a rate that is an integral multiple of the rate of the slow player (DM2). Also, the control policy of the slow player relative to the fast player is assumed to be all-digital in that the slow player applies zero or no control during those instants when the fast player is acting on the system alone.

Such situations arise in practice. For example, consider a decentralized control problem where the controllers must communicate with the system via a digital channel and the maximum

throughput rate of each channel is different. A second application of multirate descriptor games occurs in the field of agricultural economics. Here, the suppliers typically act upon the market at a slow rate (e.g., annually), whereas the consumers act upon the market at a fast rate (e.g., daily). There are other examples of problems that are more accurately modelled with multirates rather than with single rates. Moreover, there are other interesting features of multirate games that are, in general, not present in single-rate discrete-time games. For a complete, self-contained treatment of multirate Nash game theory using a *state-space* formulation, the reader is referred to [39].

1.2.2 Computational Issues

The numerical aspects of iterating coupled discrete-time Riccati equations are the topic of this thesis. The computational details are highly nontrivial because of the coupling. This fact is the source of all numerical hardships encountered in this problem. In order to produce any algorithm for solving these equations, the coupling must be removed.

Initially the relevant equations are gathered together and preliminary notation is defined. A cross-substitution procedure begins the decoupling process. Eventually the equations are rewritten in a form that permits iteration. At this point, an algorithm is presented which simultaneously iterates the Riccati equations and solves for the feedback matrices needed if the problem is motivated by a descriptor game. As given, the algorithm is designed to solve single-rate and multirate descriptor games via dynamic programming. But this fact notwithstanding, the task of iterating a set of coupled discrete-time Riccati equations is still accomplished. Furthermore, the conditions which govern the existence of a solution to the iteration problem (i.e., existence of the next Riccati iterates) become apparent and are stated formally.

However, it is necessary to study the computational aspects of iterating coupled Riccati equations in greater detail. Since these equations involve several positive-(semi)definite, symmetric matrices and several quadratic forms, it is most wise to seek an expression where these quantities appear explicitly and often. With this goal in mind, the Riccati equations are rewritten in a form

more amenable to computer implementation. Then, the specifics of the coupled Riccati algorithm are investigated. Key matrices (usually positive-(semi)definite and symmetric) are identified and an algorithm for computing each one is presented. The LINPACK software is chosen for the task of manipulating (i.e., decomposing, inverting, et cetera) these key matrices while exploiting their special structure whenever possible. Not surprisingly, a reduction in the amount of computation results from this method. These low-level algorithms are subsequently used to build the coupled Riccati package. This structured programming technique yields highly modular and efficient code. These aspects of the software engineering process make the coupled Riccati software developed here superior to other approaches.

Finally, existence and convergence issues are addressed. It is natural to ask if the coupled Riccati iterates converge to a fixed point in the limit as the number of iterations tends toward infinity. To date, neither necessary nor sufficient conditions insuring such convergence have been established. This work investigates existence issues associated with finite horizon problems and convergence issues associated with infinite horizon problems. In particular, a contraction mapping argument is developed that guarantees existence of and convergence to a unique fixed point for the infinite horizon coupled Riccati problem.

The main analysis tool used to obtain the convergence results is asymptotic analysis. First, a small parameter, ϵ_1, is introduced into one DM's cost functional. As $\epsilon_1 \rightarrow 0$, the two-player LQ descriptor Nash game problem reduces to the LQ descriptor regulator problem. Thus, an initial "bridge" between the fields of optimal control and Nash games is established. Furthermore, by setting $\epsilon_1 = 0$ periodically, a new game called multirates is created. This is the essence of the limiting argument mentioned in the last subsection. Moreover, there are several practical situations where multirate game theory is more applicable than standard single-rate game theory.

Next, a second small parameter, ϵ_2, is introduced into the other DM's cost functional. Then we let $\epsilon_1 \rightarrow 0$ and $\epsilon_2 \rightarrow 0$ independently. Subsequently, we discover that under appropriate assumptions, there exists a region where the coupled discrete-time Riccati iterations behave as a

contraction mapping. Therefore, we produce sufficient conditions that guarantee that the Riccati iterates will converge to a unique fixed point in the limit. Furthermore, empirical simulations have verified the contraction mapping behavior and indicated that the bounds obtained are rather conservative.

1.2.3 Software Implementation

The computer algorithms and software developed for solving both single and coupled discrete-time Riccati equations are integrated into the L-A-S CACSD language as new operators. This approach to software implementation is novel and unique. Theory [19,40] and software [26] have been developed for the numerical solution of single algebraic Riccati equations. Additional work that is very closely related to this problem includes [41-44]. However, until now similar efforts for coupled, discrete-time Riccati difference equations have not been undertaken. Furthermore, descriptor variable theory [19,20,43] is applied to the LQ Nash game formulation which leads to the so-called *generalized* coupled Riccati equations. Thus, the class of problems that can be solved is broadened.

Each new operator involves careful structuring of the algorithm as well as a sound software engineering basis for writing the codes. The high-level Riccati routines interface with the L-A-S protocol. The low-level routines are highly modular and computationally efficient. In the computational analysis it is shown that the task of iterating coupled Riccati equations reduces to solving systems of linear algebraic equations where one or more matrices have special properties (like positive-definite and symmetric). Hence, the decision to build the low-level routines from calls to the LINPACK library is fairly justified. Further, the high-level routines are built from frequent calls to the low-level routines as good structured programming practice dictates. The overall software structure is depicted in Figure 2.

Altogether, there are a total of six new L-A-S operators. Their mnemonic names are SYST, LQ, DRE, GAME, LQNG, and MLTR. Collectively they form a single and coupled discrete-time

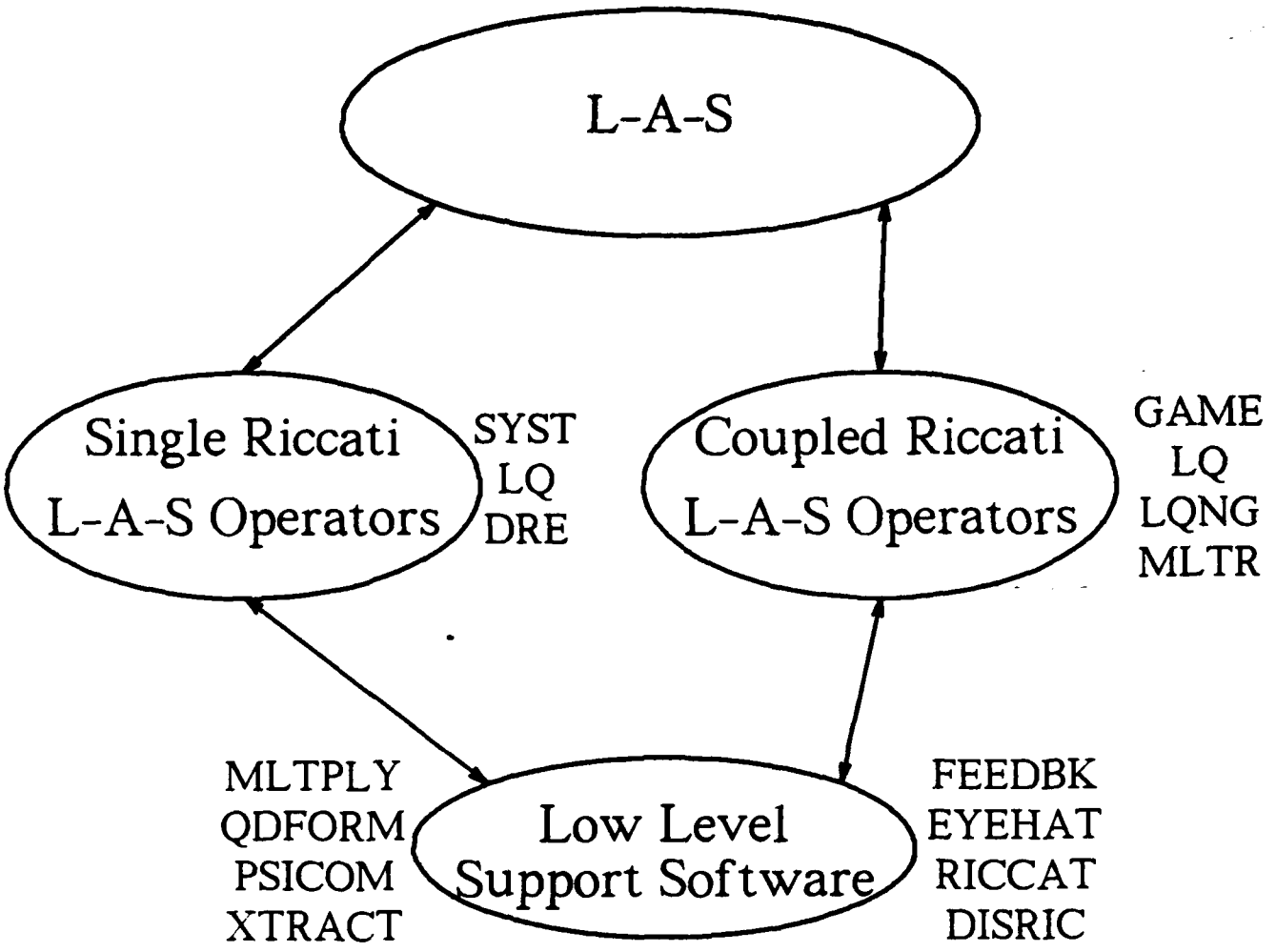


Figure 2. Coupled Riccati Software Structure and Interface with L-A-S

Riccati equation solver subpackage of L-A-S. Descriptor-variable systems are handled directly. Also, auxiliary codes are required for repeatedly performing small tasks (e.g., multiplying two general matrices). The new L-A-S operators plus the associated auxiliary support routines are written entirely in FORTRAN 77 [25] and amount to approximately 3000 lines of code.

1.3 Thesis Organization

This dissertation is organized in the following manner. Chapter 2 formulates two distinct yet related two-player, LQ dynamic games using descriptor-variable theory. The first has DMs that control the descriptor system at a single rate while the second restricts one DM to apply control at a slower rate. Assuming perfect-state information, linear state-feedback Nash strategies are determined for each case. The solution to both problems requires the iteration of two coupled discrete-time Riccati equations. The purpose of this chapter is three-fold. First, basic concepts and notation are introduced. Second, the coupled Riccati problem is motivated. Third, the theory of descriptor games and multirate games is formalized and documented.

Chapter 3 investigates the computational aspects associated with iterating two coupled discrete-time Riccati equations. A cross-substitution procedure is used to decouple the equations. A preliminary iteration algorithm is presented in a game context. Then attention is briefly directed to existence issues (iterability - the ability to iterate a recursive equation). Next, the coupled Riccati equations are rewritten in a form more amenable to computer implementation. Matrices are identified that possess special structure (e.g., positive/symmetric) and recur repeatedly throughout the equations. Algorithms for computing these matrices are given. Next, calculations of the feedbacks and the Riccati iterates are described. Then, the multirate Riccati algorithm is presented.

The existence of solutions to finite-horizon problems and convergence of Riccati iterates for infinite-horizon problems are studied in greater depth. Several new results are stated. The contraction mapping argument is developed here.

Chapter 4 is devoted to the new L-A-S operators created for solving single and coupled Riccati iterations. Details of the design and syntax of each operator are given. Also, examples illustrate how the L-A-S language may be used to study single-rate and multirate LQ descriptor Nash games. Chapter 5 summarizes this work and discusses future research topics.

Three appendices are included which support the theoretical and numerical results presented in the main body of this thesis. Appendix A contains selected software listings of the low-level

Riccati routines. Appendix B consists of facts and lemmas that are used in proving the contraction mapping result. Appendix C contains the L-A-S program run which produced the data used in the contraction mapping example of Subsection 3.4.3.

CHAPTER 2

PRELIMINARIES

This chapter formulates and solves two LQ dynamic game problems in discrete time. The first problem considers a descriptor-variable system where the DMs control the plant at the same rate. The second problem, a multirate game, further restricts one DM to apply control at a slower rate. Assuming perfect-state information, linear state-feedback Nash strategies are determined for each situation. Of more interest, however, is the fact that both problems lead to the solution of two coupled discrete-time Riccati equations.

Although the descriptor-variable formulation has been applied to the LQ regulator problem [21], it has not been attempted for dynamic games. Hence, this approach is completely new. Moreover, multirate descriptor game problems are heretofore posed yet unsolved. As such, this chapter additionally provides significant extensions to the theory of infinite dynamic games.

However, the primary purpose of the following discussion is to motivate the coupled Riccati problem whose computational aspects are studied in the remainder of this thesis. Second, basic concepts and notation are defined.

2.1 LQ Discrete–Time Descriptor Nash Games

Subsection 2.1.1 formulates an LQ descriptor game in discrete time. Subsection 2.1.2 states the Nash equilibrium solution to the proposed game.

2.1.1 Problem Formulation

Consider the linear shift-invariant discrete-time, descriptor system described by

$$E\,x(k+1) = A\,x(k) + B_1 u_1(k) + B_2 u_2(k) \qquad (k \geq 0,\ E\,x(0) = E\,x_0) \qquad (2.1.1)$$

$$y_1(k) = C_1 x(k) \qquad (2.1.2a)$$

$$y_2(k) = C_2 x(k)\ . \qquad (2.1.2b)$$

This game problem has two decision makers, DM1 and DM2. The discrete-time dynamic system is evolving at a rate indexed by the integer-valued variable, k. $x(k) \in R^n$, $u_1(k) \in R^{p_1}$, $u_2(k) \in R^{p_2}$, $y_1(k) \in R^{m_1}$, and $y_2(k) \in R^{m_2}$. The matrix E is square and is assumed to be nonsingular to numerical precision. Multiplying (2.1.1) by E^{-1} yields the standard state-space, dynamic game formulation.

Each DM must have an objective, which may or may not agree with the other DM's objective. Assume that an LQ cost functional has been selected for assessing the payoffs/losses incurred by each DM. This is frequently done in modern control/game applications. Then, the performance criterion to be minimized by each DM is given by

$$J_1(\gamma_1,\gamma_2) \triangleq \frac{1}{2} \sum_{k=0}^{T_f} y_1^T(k+1)\, S_1(k+1)\, y_1(k+1) + u_1^T(k)\, R_1(k)\, u_1(k) \tag{2.1.3a}$$

$$J_2(\gamma_1,\gamma_2) \triangleq \frac{1}{2} \sum_{k=0}^{T_f} y_2^T(k+1)\, S_2(k+1)\, y_2(k+1) + u_2^T(k)\, R_2(k)\, u_2(k) \tag{2.1.3b}$$

where γ_1, γ_2 denote the mappings from the information set to the control and T denotes transposition. The time-varying matrices $S_i(k+1)$ and $R_i(k)$, i=1,2 are symmetric and positive semidefinite. Equation (2.1.3) describes a finite-time game of duration T_f . If $T_f \rightarrow \infty$, then the infinite-time problem can be studied. Let

$$\mathbf{H} \triangleq \{0, 1, 2, \ldots, T_f\} \tag{2.1.4}$$

denote the index set of the horizon of the problem. Also, let $\mathbf{N} \triangleq \{1, 2\}$ be the set used to index the decision makers. Next, recall the definition of a Nash equilibrium for a two-player game :

***Definition* 2.1** : Nash Equilibrium [see 7 for details]

A set of strategies $\{\gamma_1^*, \gamma_2^*\}$ constitutes a Nash equilibrium solution if and only if

$$J_1^* \triangleq J_1(\gamma_1^*, \gamma_2^*) \leq J_1(\gamma_1, \gamma_2^*) \tag{2.1.5a}$$

$$J_2^* \triangleq J_2(\gamma_1^*, \gamma_2^*) \leq J_2(\gamma_1^*, \gamma_2) \tag{2.1.5b}$$

for all $\{\gamma_1, \gamma_2\} \in \Omega$ - the set of all admissible mappings. A star denotes the value of that quantity at the Nash equilibrium solution.

In this chapter, we solve for the so-called linear, state-feedback (perfect-state information) Nash equilibrium [7]. By definition, this solution obeys the principle of optimality. Since each DM will assume that the other DM will play a linear, state-feedback strategy, the feedback (perfect-state) Nash policy of each player in this game is defined by

$$u_1(k) = \gamma_1^k(x(k)) = -F_1(k)\,x(k) \tag{2.1.6a}$$

$$u_2(k) = \gamma_2^k(x(k)) = -F_2(k)\,x(k)\ . \tag{2.1.6b}$$

Then,

$$\gamma_1 = \left\{\gamma_1^0, \gamma_1^1, \ldots, \gamma_1^{T_f}\right\} \tag{2.1.7a}$$

$$\gamma_2 = \left\{\gamma_2^0, \gamma_2^1, \ldots, \gamma_2^{T_f}\right\}\ . \tag{2.1.7b}$$

We solve the basic LQ game problem described above in the next subsection.

2.1.2 Nash Equilibrium Solution of LQ Discrete–Time Descriptor Games

Define the system matrix of each DM as follows :

$$A_1(k) \stackrel{\Delta}{=} A - B_2F_2(k) \text{ and} \tag{2.1.8a}$$

$$A_2(k) \stackrel{\Delta}{=} A - B_1F_1(k)\ . \tag{2.1.8b}$$

Then utilizing (2.1.1)-(2.1.2) and (2.1.6) as well as (2.1.8), we can write (2.1.3) as :

$$J_1(\gamma_1,\gamma_2) = \frac{1}{2}\sum_{k=0}^{T_f} x^T(k)\left[A_{CL}^T(k)\,C_1^T S_1(k+1)C_1\,A_{CL}(k) + F_1^T(k)\,R_1(k)\,F_1(k)\right]x(k) \tag{2.1.9a}$$

$$J_2(\gamma_1,\gamma_2) = \frac{1}{2}\sum_{k=0}^{T_f} x^T(k)\left[A_{CL}^T(k)\,C_2^T S_2(k+1)C_2\,A_{CL}(k) + F_2^T(k)\,R_2(k)\,F_2(k)\right]x(k) \tag{2.1.9b}$$

where $A_{CL}(k) \stackrel{\Delta}{=} A_1(k) - B_1F_1(k) = A_2(k) - B_2F_2(k)$. Here the subscript CL means closed loop.

Because the feedback (perfect state) Nash equilibrium solution obeys the principle of optimality, this two-player game is equivalent to two coupled one-player LQ regulator problems. The following result concerning single-player optimization has been extended to descriptor systems. It forms the basis for the entire coupled Riccati equation study.

***Fact* 2.1** : Optimal LQ Regulator for Discrete-Time Descriptor Systems

The linear state feedback policy γ_i, $i \in \mathbf{N}$ defined in (2.1.6) satisfies (2.1.5) subject to (2.1.1)-(2.1.3) if and only if

$$u_i^*(k) = -(R_i(k) + B_i^T K_i(k+1) B_i)^{-1} (B_i^T K_i(k+1) A_i(k))\, x^*(k) \tag{2.1.10}$$

where $K_i(k)$ satisfies the following generalized discrete-time Riccati equation :

$$E^T K_i(k) E$$

$$= A_i^T(k) K_i(k+1) A_i(k) - (A_i^T(k) K_i(k+1) B_i)(\Gamma_i(k))^{-1} (B_i^T K_i(k+1) A_i(k)) + Q_i(k) \tag{2.1.11a}$$

$$= A_i^T(k) \left[K_i(k+1) - K_i(k+1) B_i (\Gamma_i(k))^{-1} B_i^T K_i(k+1) \right] A_i(k) + Q_i(k) \tag{2.1.11b}$$

$$= A_i^T(k) \left[(K_i(k+1))^{-1} + B_i (R_i(k))^{-1} B_i^T \right]^{-1} A_i(k) + Q_i(k) \tag{2.1.11c}$$

and

$$Q_i(k) \triangleq C_i^T S_i(k+1) C_i, \tag{2.1.12}$$

$$\Gamma_i(k) \triangleq R_i(k) + B_i^T K_i(k+1) B_i \tag{2.1.13}$$

with terminal constraint

$$E^T K_i(T_f + 1) E = Q_i(T_f) = C_i^T S_i(T_f + 1) C_i . \tag{2.1.14}$$

Hence,

$$F_1(k) = (R_1(k) + B_1^T K_1(k+1) B_1)^{-1} (B_1^T K_1(k+1) A_1(k)) \tag{2.1.15a}$$

$$F_2(k) = (R_2(k) + B_2^T K_2(k+1) B_2)^{-1} (B_2^T K_2(k+1) A_2(k)) . \tag{2.1.15b}$$

Proof : For the case $E = \mathbf{I}$, this result is well known. A proof of this case may be found in [7, p.221]. The extension to the case of arbitrary (and possibly singular) E has been investigated in [43,44]. However, it is illustrative to outline the major components of the proof. Let $p_i(k)$ denote the codescriptor vector for the i^{th} player, $i \in \mathbf{N}$. Then each DM faces the following two-point boundary value problem

$$\begin{bmatrix} E & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_i^T & \mathbf{0} \\ \mathbf{0} & -B_i^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} x(k+1) \\ p_i(k+1) \\ u_i(k+1) \end{bmatrix} = \begin{bmatrix} A_i & \mathbf{0} & B_i \\ -Q_i & E^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & R_i \end{bmatrix} \begin{bmatrix} x(k) \\ p_i(k) \\ u_i(k) \end{bmatrix} \tag{2.1.16}$$

with boundary conditions

$$E\,x(0) = E\,x_0 \tag{2.1.17}$$

and

$$E^T\,p_i(T_f) = E^T\,K_i(T_f)\,E\,x(T_f)\,. \tag{2.1.18}$$

Hence, the relationship between descriptor and codescriptor vectors is

$$p_i(k) = K_i(k)\,E\,x(k)\,. \tag{2.1.19}$$

In view of (2.1.5), it is well-known [7.45] that the necessary (and sufficient) conditions for existence of a minimum of (2.1.3) restricted to the class of linear state feedback policies relies upon application of the Matrix Minimum Principle (MMP) [46] to the cost functionals. Noting that, in general, the elements of F_1 and F_2 are independent, it is straightforward to take the state feedback form (2.1.9) of J_1 and J_2 in order to calculate

$$\frac{\partial\, tr\,(J_i)}{\partial F_i} = 0 \text{ implies } u_i^*(k) = -(R_i(k) + B_i^T K_i(k+1)B_i)^{-1}\,(B_i^T K_i(k+1)A_i(k))\,x^*(k)\,. \tag{2.1.20}$$

Thus, (2.1.10) and (2.1.15) are verified. The discrete-time Riccati equation (2.1.11) can be derived from (2.1.16). Let $G_i \triangleq B_i\,R_i^{-1}\,B_i^T$. Then (2.1.16) is equivalent to

$$\begin{bmatrix} E & G_i \\ 0 & A_i^T \end{bmatrix} \begin{bmatrix} x(k+1) \\ p_i(k+1) \end{bmatrix} = \begin{bmatrix} A_i & 0 \\ -Q_i & E^T \end{bmatrix} \begin{bmatrix} x(k) \\ p_i(k) \end{bmatrix}. \tag{2.1.21}$$

Therefore,

$$E\,x(k+1) + G_i\,p_i(k+1) = A_i\,x(k). \tag{2.1.22a}$$

But from (2.1.19) we have

$$\left(K_i^{-1} + G_i\right) p_i(k+1) = A_i\,x(k). \tag{2.1.22b}$$

Or,

$$A_i^T\,p_i(k+1) = A_i^T\left(K_i^{-1} + G_i\right)^{-1} A_i\,x(k) = -Q_i\,x(k) + E^T\,K_i\,E\,x(k). \tag{2.1.22c}$$

Since (2.1.22c) must be true for all $x(k)$, we require that

$$A_i^T\left(K_i^{-1} + G_i\right)^{-1} A_i + Q_i = E^T\,K_i\,E\,. \tag{2.1.22d}$$

It is clear that (2.1.11c) and (2.1.22d) are equal. The terminal condition (2.1.14) comes from setting $T_f = -1$ in (2.1.3). See [5.7.44] for details.

□

Remarks :

1) Equation (2.1.11) represents several forms of the discrete-time Riccati equation. All are equivalent assuming the appropriate inverses exist.

2) The coupling of the Riccati equations (2.1.11), i ∈ **N** occurs through the feedback matrices, $F_i(k)$. To see this, substitute (2.1.15) into (2.1.8). The resulting equations are coupled. Since (2.1.11) depends on (2.1.8) and (2.1.15), the Riccati equations are coupled too.

3) After the lengthy discussion in Chapter 1 about the advantages of descriptor-variable formulations, it is fortunate that the matrix E^{-1} appears nowhere in the derivation. However, a close examination of (2.1.11) reveals that $K_i(k)$ is not easily obtainable given the right-hand side of the equation. In fact, the case of E singular poses an interesting problem because then there may be zero, one, or multiple $K_i(k)$'s depending upon the right-hand side. Nevertheless, the Riccati algorithm presented in Chapter 3 will recover $K_i(k)$ without explicitly inverting E. Because of the complications which arise from a singular E matrix, we impose the following restriction.

Assumption **2.1** : Throughout the remainder of this thesis, the matrix E is always assumed to be nonsingular to working numerical precision.

2.2 Multirate LQ Descriptor Nash Games

This section develops the theory of multirate descriptor Nash games. The idea is that one DM is constrained to control the system at a rate which is slower than the other DM. It turns out that multirate games can be formulated and solved via single-rate Nash game theory if appropriate limiting arguments are constructed. Theorem 2.1 in this section makes the last statement precise. In addition, there are several practical situations where the optimization problem is more accurately modelled by multirates than by single rates. Chapter 1 contains those details.

The organization of this section is identical to the last one. Subsection 2.2.1 formulates the problem. Subsection 2.2.2 discusses the Nash solution of this multirate game problem. In this

thesis, the multirate theory is introduced here and mentioned in subsequent chapters as a special topic. The reader is referred to [39] for a complete, self-contained treatment of multirate game theory using a state-space formulation.

2.2.1 Problem Formulation

Consider the linear shift-invariant discrete-time system described by (2.1.1)-(2.1.2). This game problem has two decision makers, DM1 and DM2. DM1 will be referred to as the "fast" player and DM2 will be referred to as the "slow" player throughout the discussion. The discrete-time descriptor system is evolving at a rate indexed by the integer-valued variable, k. DM1 plays at rate k, i.e., at every k. However DM2 is constrained to play slower, say at rate j, where j is also an integer-valued variable and is related to k by a positive integer N as follows :

$$j \triangleq \begin{cases} k/N & \text{whenever } k/N \in \{0,1,2,\ldots\} \\ \text{undefined} & \text{otherwise} \end{cases} . \tag{2.2.1}$$

Thus, j simply indexes the state of the underlying dynamic system (2.1.1)-(2.1.2) sampled every N^{th} time. But j and k are really dummy variables. Hence, it is preferable to define a real-valued variable, $L \triangleq k/N$, for all k. Then joint interaction of DM1 and DM2 on the underlying system occurs if and only if

$$L \triangleq \frac{k}{N} \in \{0, 1, 2, \ldots\} \triangleq Z^+ . \tag{2.2.2}$$

The condition (2.2.2) will play a central role in the characterization of the solution to this multirate game problem.

The performance index to be minimized by each DM is given by (2.1.3) as before. The time-varying matrices $S_i(k+1)$ and $R_i(k)$, i=1,2 are symmetric and positive semidefinite. Equation (2.1.3) describes a finite-time game of duration T_f. If $T_f \rightarrow \infty$, then the infinite horizon multirate LQ descriptor Nash game may be studied. The definitions (2.1.6) and (2.1.7) are valid as well. Furthermore, we seek the feedback Nash equilibrium, so (2.1.8)-(2.1.9) hold. However, this Nash equilibrium will be for a *multirate* game: hence, some differences in problem formulation are to be expected.

In order to account for the fact that one DM is applying control at a slower rate, we define the all-digital control policy as follows :

All Digital Control Policy

$$u(k) \stackrel{\Delta}{=} \begin{cases} u^*(k) & \text{if } L \in Z^+ \\ 0 & \text{else (no control applied)} \end{cases} . \qquad (2.2.3)$$

Remark : In some applications, a multirate Nash game may arise where the "slow" player(s) employs a sampled-data control policy, that is, the control law is held at its previous value until k is such that the condition (2.2.2) holds. Such problems will be addressed elsewhere.

We solve the basic multirate, LQ game problem described above in the next subsection. In doing so, we will discuss the meaning of a Nash equilibrium in a *multirate* setting.

2.2.2 Nash Equilibrium Solution of Multirate Descriptor Games

In this subsection, we show that under appropriate assumptions, the multirate game defined in Subsection 2.2.1 can be solved using standard single-rate game theory with minor modifications. The discussion and results of Subsection 2.1.2 are valid here. Now, we construct a limiting argument which shows that as the small parameter $\epsilon \to 0$, the all-digital control policy (2.2.3) is achieved asymptotically.

For notational reasons, let us study a slightly simplified version of the all-digital multirate game formulated in the previous subsection. Consider the weighting matrices as an ordered pair. Then, we are interested in the quantities $(S_1(k+1), R_1(k))$ and $(S_2(k+1), R_2(k))$.

Define :

$$(S_1(k+1), R_1(k)) \equiv (S_1, R_1) \text{ where } S_1 \geqslant 0 \text{ and } R_1 > 0 \text{ are constant matrices for all k.} \qquad (2.2.4)$$

and

$$(S_2(k+1), R_2(k)) \stackrel{\Delta}{=} \begin{cases} (S_2, R_2) \text{ where } S_2 \geqslant 0 \text{ and } R_2 > 0 & \text{when } L \in Z^+ \\ (\mathbf{0}, \frac{1}{\epsilon}\mathbf{I}) \text{ where } \epsilon \to 0 & \text{otherwise} \end{cases} . \qquad (2.2.5)$$

Here, $\mathbf{0}$ is the zero matrix and $\mathbf{I}$ is the identity matrix.

The main results are equally applicable to arbitrary but bounded $S_1(k+1)$, $S_2(k+1)$, $R_1(k)$, and $R_2(k)$, (when $L \in \mathbf{Z}^+$). Furthermore, this setup accounts for the fact that DM2 is applying control at a slower rate. In fact, as $\epsilon \to 0$, the optimal control of DM2 approaches, in the limit, the all-digital control policy described by (2.2.3). Furthermore, under mild boundedness assumptions, both J_1 and J_2 are finite. Hence, the problem is well-posed. These last statements are supported by the next result.

Theorem **2.1** : Characterization of Multirate Nash Games

For the pair $(S_2(k+1), R_2(k))$ as defined in (2.2.5), if $A_i(k)$ and $K_i(k+1)$, i=1,2 remain bounded for all $k \in \mathbf{H}$, then as $\epsilon \to 0$, $\| u_2^*(k) \| \to 0$ whenever $L \notin \mathbf{Z}^+$. Further, whenever $L \notin \mathbf{Z}^+$, $\| \Delta J_2^*(k) \| \to 0$ where

$$\Delta J_i(k) \triangleq y_i^T(k+1) S_i(k+1) y_i(k+1) + u_i^T(k) R_i(k) u_i(k) \ . \tag{2.2.6}$$

Moreover, $u_2^*(k)$ is $O(\epsilon)$ and so is $\Delta J_2^*(k)$.

Proof : Throughout the proof all norms taken are assumed to be the standard Euclidean norm. Also, it is useful to define :

$\bar{\sigma}(\cdot) \triangleq$ largest singular value of $(\cdot) = \| \cdot \|$, and

$\underline{\sigma}(\cdot) \triangleq$ smallest singular value of $(\cdot)$.

It is given in the problem statement that B_1, B_2, S_1, S_2, and R_1 are matrices with bounded entries. Now as $\epsilon \to 0$, $\| R_2(k) \| = \frac{1}{\epsilon} \to \infty$ whenever $L \notin \mathbf{Z}^+$. In fact, $\underline{\sigma}(R_2(k)) = \bar{\sigma}(R_2(k)) = \frac{1}{\epsilon}$ by construction. For the moment, assume that $K_1(k+1)$, $K_2(k+1)$, $A_1(k)$, and $A_2(k)$ exist and are finite. This requirement is investigated further in the next chapter. Consequently, for sufficiently small ϵ, $\| \Gamma_2(k) \| \approx \frac{1}{\epsilon} \to \infty$. Specifically, from the relationship

$$\| (X)^{-1} \| = \bar{\sigma}((X)^{-1}) = \frac{1}{\underline{\sigma}(X)} \quad \text{for any matrix } X \text{ where } \underline{\sigma}(X) \neq 0 . \tag{2.2.7}$$

and the fact that for any two positive semidefinite symmetric matrices X, Y,

$$\| (X+Y)^{-1} \| = \frac{1}{\underline{\sigma}(X+Y)} \leq \frac{1}{\underline{\sigma}(X)} = \| (X)^{-1} \| \tag{2.2.8}$$

we conclude that

$$\| \left[\Gamma_2(k)\right]^{-1} \| = \left| \underline{\sigma}(\Gamma_2(k)) \right|^{-1}$$

$$\leq \left| \underline{\sigma}(R_2(k)) \right|^{-1} = \left| \frac{1}{\epsilon} \right|^{-1} = \epsilon \to 0 .$$

Hence,

$$\| u_2^*(k) \| = \| -(R_2(k) + B_2^T K_2(k+1) B_2)^{-1} \ (B_2^T K_2(k+1) A_2(k)) \ x^*(k) \| \tag{2.2.9a}$$

$$= \| -(\Gamma_2(k))^{-1} \ (B_2^T K_2(k+1) A_2(k)) \ x^*(k) \| \tag{2.2.9b}$$

$$\leq \| -(\Gamma_2(k))^{-1} \| \cdot \| B_2^T K_2(k+1) A_2(k) \| \cdot \| x^*(k) \| \tag{2.2.9c}$$

$$= \epsilon \ \| B_2^T K_2(k+1) A_2(k) \| \cdot \| x^*(k) \| \to 0 \quad \text{for all } \| x^*(k) \| < \infty . \tag{2.2.9d}$$

Thus, $u_2^*(k)$ is $O(\epsilon)$. That is, there exist non-negative constants, M_1 and M_2, such that $M_1 \epsilon \leq \| u_2^*(k) \| \leq M_2 \epsilon$. For example, choose $M_1 = 0$ and $M_2 = \| (B_2^T K_2(k+1) A_2(k)) \ x^*(k) \|$. Obviously, $R_2(k)$ and $\Gamma_2(k)$ are $O\left(\frac{1}{\epsilon}\right)$.

Having established that $u_2^*(k)$ is $O(\epsilon)$, we show that $\Delta J_2^*(k)$ is also. When $L \notin \mathbf{Z}^+$, we have from (2.2.5) and (2.2.6) that

$$\Delta J_2^*(k) = u_2^{T*}(k) \ R_2(k) \ u_2^*(k) . \tag{2.2.10}$$

Clearly, $\Delta J_2^*(k)$ is $O(\epsilon) \ O\left(\frac{1}{\epsilon}\right) \ O(\epsilon) = O(\epsilon)$.

□

Remark : From Theorem 2.1, we conclude that $\lim_{\epsilon \to 0} \| \left[\Gamma_2(k)\right]^{-1} \| = 0$ whenever $L \notin \mathbf{Z}^+$. Hence the discrete-time Riccati equation (2.1.11a) reduces to the discrete-time Lyapunov equation :

$$E^T \ K_2(k) \ E = A_2^T(k) K_2(k+1) A_2(k) + Q_2(k) .$$

Therefore, as $\epsilon \to 0$, $\| u_2^*(k) \|$ as well as $\| \Delta J_2^*(k) \| \to 0$, and the all-digital control policy of (2.2.3) is realized in the limit. Hence, the multirate descriptor Nash game problem proposed in

Subsection 2.2.1 can be completely characterized using single-rate theory, if the pair $(S_2(k+1), R_2(k)) = (0, \frac{1}{\epsilon}I)$ and $\epsilon \to 0$ whenever $L \notin Z^+$. Indeed, the multirate Riccati iterations are *decoupled* when $L \notin Z^+$. But when $L \in Z^+$, the coupling is present as in the first problem of Section 2.1. Thus, multirate games require the *periodic* iteration of coupled discrete-time Riccati equations.

2.3 A Numerical Example

In this section, a numerical example is constructed which illustrates that there are situations where a descriptor-variable game formulation is computationally superior to the corresponding state-space game formulation in terms of numerical accuracy. Consider the system (2.1.1)-(2.1.2) with performance indices (2.1.3). Let $C_1 = C_2 = R_1 = R_2 = I$. Then choose

$$A = \begin{vmatrix} 0.5 & -1.0 & -1.0 \\ 0.0 & 0.77 & -1.0 \\ 0.0 & 0.0 & 0.001 \end{vmatrix}, \quad B_1 = \begin{vmatrix} 9.87e\,2 \\ 1.23 \\ -1.01e-3 \end{vmatrix}, \quad B_2 = \begin{vmatrix} 1.37 \\ -1.0e-3 \\ 1.0e-5 \end{vmatrix},$$

and

$$E = \begin{vmatrix} 1.0e\,0 & 1.0e\,3 & 1.0e\,6 \\ 0.0 & 9.9e\,5 & 1.0e\,9 \\ -1.95677e-1 & 0.0 & 1.0e\,15 \end{vmatrix}.$$

Notice the wide range of magnitudes in E. As a descriptor game formulation, the scalings are confined to the matrix E. However, if a state-space formulation is sought, the extreme separation of magnitudes is spread throughout the matrices A, B_1, and B_2. This is precisely the case which degrades the numerical accuracy of the final results. Next, let

$$S_1 = \begin{vmatrix} 1.2345 & 1.98752e\,3 & 9.0e\,6 \\ 1.98752e\,3 & 1.0e\,12 & 1.0e\,15 \\ 9.0e\,6 & 1.0e\,15 & 1.10102e\,38 \end{vmatrix} \quad \text{and} \quad S_2 = \begin{vmatrix} 1.94731 & 1.566e\,3 & 1.0e\,6 \\ 1.566e\,3 & 1.0e\,12 & 1.0e\,15 \\ 1.0e\,6 & 1.0e\,15 & 1.23976e\,35 \end{vmatrix}.$$

The weighting matrices S_1 and S_2 also contain elements with a wide range of scalings.

It is customary to employ single precision arithmetic when demonstrating numerical difficulties. The effect is to show how serious errors can result from even low-order problems. Assuming single precision (8 significant digits of accuracy), the feedbacks are computed using the

methods described in the thesis. The result is

$$F_1 = [\,5.06829e-4 \;\; -1.01567e-3 \;\; -9.86917e-4]$$

$$F_2 = [\,-1.73157e-4 \;\; 1.22620e-3 \;\; -1.44510e-2]\,.$$

Now, multiply equation (2.1.1) by E^{-1}. This yields three new matrices :

$$\overline{A} = \begin{vmatrix} 0.5 & -1.00078 & -0.99899 \\ -9.88268e-14 & 7.77778e-7 & -1.0101e-6 \\ 9.78385e-17 & -1.95829e-16 & -1.94479e-16 \end{vmatrix},$$

$$\overline{B}_1 = \begin{vmatrix} 9.86999e\,2 \\ 1.24223e-6 \\ 1.93132e-13 \end{vmatrix}, \quad \text{and} \quad \overline{B}_2 = \begin{vmatrix} 1.37000 \\ -1.01037e-9 \\ 2.68088e-16 \end{vmatrix}.$$

Although $\overline{E} = \mathbf{I}$, the matrices $\overline{A}$, $\overline{B}_1$, and $\overline{B}_2$ are now very poorly conditioned. Assuming the same weighting matrices, we again compute the feedbacks using the best methods available. The result is

$$F_1 = [\,5.07253e-4 \;\; -1.01652e-3 \;\; -9.87737e-4]$$

$$F_2 = [\,-4.78691e-4 \;\; 1.84028e-3 \;\; -1.3860e-2]\,.$$

Obviously, there is quite a difference - especially in the first element of F_2 where there is well over a factor of two difference. In order to get an accurate appraisal of the deviations, consider the descriptor system and run the feedback calculation using double precision arithmetic (16 significant digits of accuracy). The result is

$$F_1 = [\,5.06600e-4 \;\; -1.01521e-3 \;\; -9.86498e-4]$$

$$F_2 = [\,-7.68264e-6 \;\; 8.94645e-4 \;\; -1.47528e-2]\,.$$

The reduced precision coupled with the scalings present in the matrix E are responsible for the decreased accuracy. Assuming this last set of values to be most accurate, we compute a percent difference which is summarized in Table 2.1.

The interesting item to note for this example is that the state-space formulation always yields numbers which are three times less accurate in terms of percent difference from true value. One is led to conclude that the descriptor formulation is more numerically accurate (robust) for game problems with nearly singular E matrices. Clearly, if E is singular, the state-space formulation is not even applicable. Also, if E is sparse, then multiplying by E^{-1} is undesirable because sparsity

Table 2.1. Percent Difference Calculation

	Component 1	Component 2	Component 3
F_1 Descriptor	0.045%	0.004%	0.042%
F_1 State-Space	0.13%	0.13%	0.125%
F_2 Descriptor	2154%	37%	2%
F_2 State-Space	6131%	106%	6%

is destroyed.

CHAPTER 3

COMPUTATIONAL ASPECTS OF COUPLED DISCRETE-TIME RICCATI EQUATIONS

This chapter investigates the computational aspects of iterating the coupled discrete-time Riccati equations presented in the last chapter. It was shown there that coupled Riccati equations solve both single-rate and multirate descriptor Nash game problems. Though these games have been solved in theory, the computational details of iterating coupled Riccati equations are highly nontrivial.

As pointed out in Remark 2 following Fact 2.1, the Riccati equations are coupled through the feedback matrices. That is, the ability to iterate the Riccati equations hinges upon the knowledge of both feedbacks. But the calculation of one DM's feedback requires the knowledge of the other DM's feedback (substitute (2.1.8) into (2.1.15)). This fact is the source of all numerical hardships encountered when attempting to iterate coupled discrete-time Riccati equations. In order to produce any algorithm for iterating the Riccati equations, the coupling must be removed.

Towards this end, Section 3.1 describes a feedback decoupling procedure. The equations of Chapter 2 are manipulated in such a way that the coupling vanishes. Then an algorithm is stated which solves for the feedback matrices and iterates the Riccati equations. Hence both Nash game problems posed in the last chapter are solved numerically via dynamic programming. More important, however, is the fact that the task of iterating two coupled discrete-time Riccati equations is accomplished. Additionally, more notation is introduced. The section is concluded with an existence analysis of the algorithm. Conditions are stated which insure that the Riccati equations can be iterated.

Section 3.2 presents the computationally superior coupled discrete-time Riccati iteration algorithm. To begin with, all relevant equations are assembled together. Assuming the feedback matrices are known, the problem of efficiently iterating a Riccati equation is resolved by considering numerous forms of the equation. Thus the final choice is well-justified. Then it is observed that the terms in the Riccati and feedback expressions are composed of positive-

(semi)definite, symmetric matrices and quadratic forms. Therefore, algorithms are devised for computing these frequently-used quantities. Also, routines for multiplying general matrices and matrices with special structure are given.

The task of computing the feedback matrices is addressed next. Unfortunately, the (implicit) inversion of one general square matrix is required. In fact, one of two square matrices must be inverted. Hence, the computational burden is heavy at this stage because the condition number of each matrix must be estimated. Furthermore, matrices with special structure (e.g., symmetric) are less prevalent in the feedback expressions so the computational intensiveness problem is exacerbated. Nevertheless, it is possible to rewrite the feedback equations in a form that exposes additional structure thus alleviating the computational complexity a bit. At this juncture, the feedback algorithm is presented. Finally, the coupled Riccati algorithm is built from the various low-level algorithms defined earlier in the discussion.

Section 3.3 studies the existence of solutions to finite-horizon problems. It is here that previous results (in Section 3.1) are strengthened. Section 3.4 investigates the convergence behavior of Riccati iterations for infinite-horizon problems. The existence of a region where the coupled Riccati equations constitute a contraction mapping is established. The bulk of the section is devoted to proving the contraction. The contraction mapping argument guarantees existence of and convergence to a unique fixed point for an infinite number of Riccati iterations.

3.1 A Decoupling Procedure

This section considers some preliminary computational aspects of iterating the coupled discrete-time Riccati equations obtained in the previous chapter. First, the Riccati equations are decoupled. Then an LQ Nash game algorithm is presented. Finally, various necessary and sufficient conditions which govern existence and uniqueness of solutions to finite-horizon LQ Nash game problems (and hence coupled Riccati iterations) are stated.

Beginning with (2.1.6) use (2.1.8), (2.1.10) and (2.1.13) to obtain :

$$u_1^*(k) = -F_1(k)\,x^*(k) = -(\Gamma_1(k))^{-1}\,[B_1^T K_1(k+1)\,(A - B_2F_2(k))]\,x^*(k) \qquad (3.1.1)$$

and

$$u_2^*(k) = -F_2(k)\,x^*(k) = -(\Gamma_2(k))^{-1}\,[B_2^T K_2(k+1)\,(A - B_1F_1(k))]\,x^*(k)\ . \qquad (3.1.2)$$

Notice that (3.1.1)-(3.1.2) are coupled equations where the coupling occurs through the feedback matrices $F_1(k)$ and $F_2(k)$. In order to present an algorithm for solving the Riccati iteration problem, this coupling must be addressed. Towards this end, we will use a cross-substitution procedure to derive the functional relationships :

$$F_1(k) = g_1(K_1(k+1), K_2(k+1)) \qquad (3.1.3a)$$

$$F_2(k) = g_2(K_1(k+1), K_2(k+1))\ . \qquad (3.1.3b)$$

To simplify notation, we introduce

$$\Psi_i(k) \triangleq (\Gamma_i(k))^{-1}\,B_i^T\,K_i(k+1),\ i=1,2. \qquad (3.1.4)$$

Therefore, from (3.1.1) and (3.1.2) we conclude that :

$$F_1(k) = (\Gamma_1(k))^{-1}\,\left|\,B_1^T K_1(k+1)\,(A - B_2F_2(k))\,\right| \qquad (3.1.5a)$$

$$= \Psi_1(k)A - \Psi_1(k)B_2F_2(k) \qquad (3.1.5b)$$

$$= \Psi_1(k)A - \Psi_1(k)B_2\left|\,(\Gamma_2(k))^{-1}\,B_2^T K_2(k+1)\,(A - B_1F_1(k))\,\right| \qquad (3.1.5c)$$

$$= \Psi_1(k)\left|\,I - B_2\Psi_2(k)\,\right|A + \Psi_1(k)B_2\Psi_2(k)B_1F_1(k)\ . \qquad (3.1.5d)$$

Hence,

$$\left|\,I - \Psi_1(k)B_2\Psi_2(k)B_1\,\right|F_1(k) = \Psi_1(k)\left|\,I - B_2\Psi_2(k)\,\right|A\ . \qquad (3.1.6a)$$

Similarly, for the feedback, $F_2(k)$, we get :

$$\left|\,I - \Psi_2(k)B_1\Psi_1(k)B_2\,\right|F_2(k) = \Psi_2(k)\left|\,I - B_1\Psi_1(k)\,\right|A\ . \qquad (3.1.6b)$$

Define,

$$\Xi_1(k) \triangleq I - \Psi_1(k)B_2\Psi_2(k)B_1 \qquad (3.1.7a)$$

$$\Xi_2(k) \triangleq I - \Psi_2(k)B_1\Psi_1(k)B_2\ . \qquad (3.1.7b)$$

Then the functions g_1 and g_2 described in (3.1.3) are given by

$$F_1(k) = \left[\Xi_1(k)\right]^{-1} \Psi_1(k) \left[I - B_2\Psi_2(k)\right] A \tag{3.1.8a}$$

$$F_2(k) = \left[\Xi_2(k)\right]^{-1} \Psi_2(k) \left[I - B_1\Psi_1(k)\right] A \tag{3.1.8b}$$

assuming that $\left[\Xi_1(k)\right]^{-1}$ and $\left[\Xi_2(k)\right]^{-1}$ exist. Given the definitions of the weighting matrices, it is clear that $\left[\Gamma_1(k)\right]^{-1}$ and $\left[\Gamma_2(k)\right]^{-1}$ always exist if $K_1(k+1)$ and $K_2(k+1)$ are matrices with bounded entries. Existence of $\left[\Xi_i(k)\right]^{-1}$ i=1,2 is a more delicate issue that will be dealt with shortly.

We may now present an algorithm for solving the LQ descriptor Nash game problem. Dynamic programming [5] dictates that the feedbacks must be solved for in reverse time. The requirement of iterating two coupled discrete-time Riccati equations is automatically satisfied.

LQ Descriptor Nash Game Algorithm

Step 1 : Initializations
Set $COUNT \leftarrow T_f$.
Set $E^T K_i(COUNT+1) E = C_i^T S_i(COUNT+1) C_i$, i=1,2. (Terminal Constraint)

Step 2 : Beginning of Main Loop
Compute $\left[\Gamma_i(COUNT)\right]^{-1}$, i=1,2 using (2.1.13).

Step 3 : Core Calculations
Compute $\Psi_i(COUNT)$, i=1,2 using (3.1.4).
Compute $\Xi_i(COUNT)$, i=1,2 using (3.1.7).
If $\left[\Xi_1(COUNT)\right]^{-1}$ exists, then compute it, otherwise go to Step 5.
If $\left[\Xi_2(COUNT)\right]^{-1}$ exists, then compute it, otherwise go to Step 5.
Compute $F_i(COUNT)$, i=1,2 using (3.1.8).
Compute $A_i(COUNT)$, i=1,2 using (2.1.8).

Step 4 : Update Data
Store $F_i(COUNT)$, i=1,2 for this value of $COUNT$.
Iterate the coupled discrete-time Riccati equations according to (2.1.11) with i=1,2 thus obtaining $K_i(COUNT)$ from $K_i(COUNT+1)$, i=1,2.
Set $COUNT \leftarrow COUNT - 1$.
If $COUNT \geq 0$, then go to Step 2, otherwise STOP the algorithm.

Step 5 : Error Condition
Print an Error Message that indicates that one or more of the feedbacks are infinite at this stage of the game. Then STOP the algorithm.

Remark : Multirate LQ Descriptor Nash Game Algorithm

The corresponding multirate algorithm may be obtained by simply replacing step 2 with

Compute $\left[\Gamma_i(COUNT)\right]^{-1}$, i=1,2 using (2.1.13). If $L \notin \mathbf{Z}^+$, then Set $\left[\Gamma_2(COUNT)\right]^{-1} = \mathbf{0}$. Moreover, other simplifications are possible for a multirate game. If $L \notin \mathbf{Z}^+$, then $\left[\Gamma_2(COUNT)\right]^{-1} = 0$ which implies that $\Psi_2(COUNT) = 0$. Furthermore, (3.1.7) implies that $\Xi_1(COUNT) = \Xi_2(COUNT) = \mathbf{I}$ which, in turn, implies that $F_1(COUNT) = \Psi_1(COUNT)A$ and $F_2(COUNT) = \mathbf{0}$. Hence, the Riccati equations that have to be iterated reduce to

$$E^T K_1(k) E = A^T \left[K_1(k+1) - K_1(k+1)B_1(\Gamma_1(k))^{-1}B_1^T K_1(k+1) \right] A + Q_1(k)$$

$$E^T K_2(k) E = A_2^T(k) K_2(k+1) A_2(k) + Q_2(k) .$$

Many facts become apparent from the descriptor Nash game algorithm. They are summarized by the following :

Proposition **3.1** : Existence of Feedbacks

$K_1(k)$ exists if and only if $\| \left[\Xi_1(k)\right]^{-1} \| < \infty$. Likewise, $K_2(k)$ exists if and only if $\| \left[\Xi_2(k)\right]^{-1} \| < \infty$. Equivalently, there exists a unique solution $(\gamma_1^{k^*}, \gamma_2^{k^*})$, $k \in \mathbf{H}$ to the k^{th} stage of the LQ descriptor Nash game described in Chapter 2 if and only if $\underline{\sigma}(\Xi_1(k)) \neq 0$ and $\underline{\sigma}(\Xi_2(k)) \neq 0$ for that k. Therefore, there exists a unique solution policy (γ_1^*, γ_2^*) to the entire Nash game if and only if $0 \notin \{ \underline{\sigma}(\Xi_i(k) \mid k \in \mathbf{H}, i=1,2\}$.

Proof : Consider only single-rate games and let $k=T_f$. Then, $K_i(k+1)$, i=1,2 are well-defined by (2.1.14). Assumption 2.1 guarantees that $K_i(k+1)$ is unique. Since, $0 < R_1 < \infty$ and $\| B_1 \| < \infty$, $\left[\Gamma_1(k)\right]^{-1}$ exists. Similarly, $0 < R_2 < \infty$ and $\| B_2 \| < \infty$ implies that $\left[\Gamma_2(k)\right]^{-1}$ exists. Thus, $\left[\Gamma_i(k)\right]^{-1}$, i=1,2 exists if and only if $K_i(k+1)$ exists, respectively.

Now, given that $\left[\Gamma_i(k)\right]^{-1}$, i=1,2 exists, then by (3.1.4) so does $\Psi_i(k)$. Furthermore, from (3.1.8), it is clear that $F_i(k)$, i=1,2 exists if and only if the corresponding $\left[\Xi_i(k)\right]^{-1}$ exists. This condition is the same as requiring $\underline{\sigma}(\Xi_i(k)) \neq 0$, i=1,2. From (2.1.11a), it is obvious that boundedness of $K_i(k)$ hinges upon boundedness of $A_i(k)$ which, from (2.1.8), occurs if and only if $\| F_i(k) \| < \infty$, i=1,2. Therefore, $K_i(k)$ exists if and only if $\overline{\sigma}\left[(\Xi_i(k))^{-1}\right] = \dfrac{1}{\underline{\sigma}(\Xi_i(k))} < \infty$. That is, $\underline{\sigma}(\Xi_i(k)) \neq 0$, i=1,2. Uniqueness comes from the fact that (2.1.11a) defines a single matrix, $E^T K_i(k) E$ given all quantities on the right-hand side of (2.1.11a) and Assumption 2.1. Finally, the proof is completed by inductively applying the above argument to the next (i.e., $k-1^{st}$) stage of the game as long as $\{K_i(k), i=1,2\}$ exists. If not, then the given LQ descriptor Nash game is ill-posed because either $u_1^*(k)=\infty$ or $u_2^*(k)=\infty$ or both for some $k \in \mathbf{H}$.

□

Remark : Existence of Multirate Feedbacks

If $L \in \mathbf{Z}^+$, then the result of Proposition 3.1 applies. However, whenever $L \notin \mathbf{Z}^+$, the remark following the LQ descriptor Nash game algorithm indicates that both feedbacks always exist. Notice that if $N=2$, then the conditions $L \in \mathbf{Z}^+$ and $L \notin \mathbf{Z}^+$ occur with equal frequency. But if $N>2$, then $L \notin \mathbf{Z}^+$ occurs more often. Hence, as N increases, the frequency with which the feedbacks are guaranteed to exist increases.

As an immediate consequence of Proposition 3.1, we have :

Corollary **3.1** : A sufficient condition for existence of a unique solution to the finite-horizon LQ Nash game problem is that $\overline{\sigma}(\Psi_1(k)B_2\Psi_2(k)B_1) < 1$ and $\overline{\sigma}(\Psi_2(k)B_1\Psi_1(k)B_2) < 1$ for all $k \in \mathbf{H}$.

Proof : It is well known [18,47] that the singular values of a matrix are intimately related to the problem of rank degeneracy. In particular, given a square matrix X with $\underline{\sigma}(X) \triangleq \underline{\sigma}_X > 0$ and another square matrix Y of compatible dimension with $\overline{\sigma}(Y) \triangleq \overline{\sigma}_Y \geq 0$, then if $\overline{\sigma}_Y < \underline{\sigma}_X$, the

matrix $(X+Y)$ has full rank. Consider (3.1.7a) and take $X = \mathbf{I}$ and $Y = \Psi_1(k)B_2\Psi_2(k)B_1$. Obviously, $\overline{\sigma}(\mathbf{I}) = \underline{\sigma}(\mathbf{I}) = 1$. Hence as long as $\overline{\sigma}(Y) = \overline{\sigma}(\Psi_1(k)B_2\Psi_2(k)B_1) < 1$, the matrix $\Xi_1(k)$ can never be singular. Thus, $\left[\Xi_1(k)\right]^{-1}$ always exists. Similarly, consider (3.1.7b) and take $Y = \Psi_2(k)B_1\Psi_1(k)B_2$ leaving X as before. The same singular value argument can be applied to this case, which then proves the Corollary.

□

***Corollary* 3.2 :** Another sufficient condition for existence of a unique solution to the finite-horizon LQ Nash game problem is that $\underline{\sigma}(\Psi_1(k)B_2\Psi_2(k)B_1) > 1$ and $\underline{\sigma}(\Psi_2(k)B_1\Psi_1(k)B_2) > 1$ for all $k \in \mathbf{H}$.

***Proof* :** Using the same singular value argument as discussed in Corollary 3.1, consider (3.1.7a) and take $Y = \mathbf{I}$ and $X = \Psi_1(k)B_2\Psi_2(k)B_1$. As long as $\underline{\sigma}(X) > \overline{\sigma}(Y) = 1$, the matrix $\Xi_1(k)$ can never be singular. Similarly, consider (3.1.7b) and take $X = \Psi_2(k)B_1\Psi_1(k)B_2$ leaving Y as before. The same singular value argument can be applied to this case, which then proves the Corollary.

□

We can carry these last two corollaries one step further by restricting attention to those games in which both players have a single input to the system. In this case, define the scalars :

$$\omega_1(k) \triangleq \Psi_1(k)B_2$$

$$\omega_2(k) \triangleq \Psi_2(k)B_1 \ .$$

Then, $\Xi_1(k) = 1 - \omega_1(k)\omega_2(k) = 1 - \omega_2(k)\omega_1(k) = \Xi_2(k)$. Hence,

***Corollary* 3.3 :** A unique solution to the finite horizon LQ Nash game problem with single inputs for each player exists if and only if $\omega_1(k) \neq (\omega_2(k))^{-1}$ for all $k \in \mathbf{H}$.

***Proof* :** Obviously, $\Xi_1(k) = \Xi_2(k)$ are scalar quantities for the situation when each player has a single input. Therefore, singularity of both occurs if and only if

$\omega_1(k)\omega_2(k) = \omega_2(k)\omega_1(k) = 1$

This condition is equivalent to

$\omega_1(k) = (\omega_2(k))^{-1}$ or $\omega_2(k) = (\omega_1(k))^{-1}$.

□

In summary, these results provide on-line checkable conditions so that the descriptor-game algorithm can monitor itself and determine if a Nash solution strategy exists. In the next section, algorithms are developed for iterating coupled discrete-time Riccati equations in a computationally efficient manner.

3.2 The Coupled Discrete–Time Riccati Algorithm

In Section 3.1 an algorithm is presented for solving LQ descriptor Nash games. The process involves iterating coupled discrete-time Riccati equations. This section describes a more efficient and numerically robust procedure for performing one iteration of two coupled, discrete-time Riccati equations. As a byproduct of the analysis, we obtain an equally efficient procedure for iterating a single discrete-time Riccati equation which arises in the study of the optimal LQ regulator problem.

3.2.1 Algorithm Implementation

This subsection reviews the pertinent equations involved in iterating two coupled discrete-time Riccati equations. A complete discussion of the equations may be found in Section 3.1. Because positive-(semi)definite symmetric matrices are frequently encountered in this coupled Riccati problem, the Cholesky factorization [17,24] will be used to expedite the calculations. Recall that any positive-(semi)definite symmetric matrix, A , may be factored as $A = X_A^T X_A$ where X_A is upper triangular. It is preferable to work with X_A rather than directly with A because of the triangular structure. For example, it is often necessary to calculate $C = A^{-1} B$ where B is some other compatibly-dimensioned matrix. This can be accomplished by solving the symmetric linear

system $A\ C = B$ for C. However, if the Cholesky factor of A is available then the linear system is equivalent to solving two triangular systems $X_A^T\ D = B$ and $X_A\ C = D$. This technique has two major advantages over working directly with the original system $A\ C = B$. First, triangular systems are extremely easy to solve. Second, the inverse of A is never *explicitly* computed. As a general rule of thumb, the explicit computation of an inverse should be avoided at almost any cost. For the case where A is dense and has no special structure, the Cholesky factorization is replaced by the LU factorization. In this factorization, U is upper triangular and L is the product of elementary lower triangular and permutation matrices.

Remarks : Software Engineering

The solution of this coupled Riccati problem requires software designed to handle linear algebraic equations. Frequently, positive-(semi)definite symmetric matrices appear which require special handling. The LINPACK [24] FORTRAN library is the best known and the most widely recommended [21] software package used for these situations. For all the algorithms which follow, Single (S) precision arithmetic is assumed. Therefore, only subroutines beginning with 'S' are referenced from the LINPACK library. If Double (D) precision is used, then the first letter of the corresponding LINPACK routine is 'D'. However, L-A-S is a *double precision* package. Since the coupled Riccati software will be integrated into L-A-S, the software listings in Appendix A refer to the double precision versions of LINPACK. Second, all of the Riccati software discussed here is coded in FORTRAN. This decision is primarily motivated by the fact that both L-A-S and LINPACK are FORTRAN-based packages. In addition, FORTRAN is generally chosen for coding numerical software.

Several of the algorithms require at least one multiplication of two matrices with no special structure (like upper/lower triangular). For this reason, we define algorithm MLTPLY, which multiplies two arbitrary, but compatible matrices. In particular given the matrices A and B, MLTPLY is useful for computing the forms $A\ B$ and $A^T\ B$. Notice that the case where both A and B are symmetric offers no advantages since the resulting matrix is, in general, not symmetric.

Hence, algorithm MLTPLY is applicable. Since MLTPLY is too simple to be given as a sequence of steps, a copy of the FORTRAN source code is included in Appendix A.

Now the equations needed to iterate coupled Riccatis are reviewed. The Riccati equation for Decision Maker i (DMi), $i \in \mathbf{N} \triangleq \{1, 2\}$ at stage k is given by

$$E^T K_i(k) E$$

$$= A_i^T(k) K_i(k+1) A_i(k) - A_i^T(k) K_i(k+1) B_i (\Gamma_i(k))^{-1} B_i^T K_i(k+1) A_i(k) + Q_i(k) \qquad (3.2.1)$$

where

$$\Gamma_i(k) \triangleq R_i(k) + B_i^T K_i(k+1) B_i . \qquad (3.2.2)$$

$$A_1(k) \triangleq A - B_2 F_2(k) \text{ . and} \qquad (3.2.3a)$$

$$A_2(k) \triangleq A - B_1 F_1(k) . \qquad (3.2.3b)$$

The feedback employed by each DM is

$$F_1(k) = (R_1(k) + B_1^T K_1(k+1) B_1)^{-1} (B_1^T K_1(k+1) A_1(k)) \qquad (3.2.4a)$$

$$= (\Gamma_1(k))^{-1} [B_1^T K_1(k+1) (A - B_2 F_2(k))] \qquad (3.2.4b)$$

$$= \left(\Xi_1(k)\right)^{-1} \Psi_1(k) \left| I - B_2 \Psi_2(k) \right| A \qquad (3.2.4c)$$

and

$$F_2(k) = (R_2(k) + B_2^T K_2(k+1) B_2)^{-1} (B_2^T K_2(k+1) A_2(k)) \qquad (3.2.5a)$$

$$= (\Gamma_2(k))^{-1} [B_2^T K_2(k+1) (A - B_1 F_1(k))] \qquad (3.2.5b)$$

$$= \left(\Xi_2(k)\right)^{-1} \Psi_2(k) \left| I - B_1 \Psi_1(k) \right| A \qquad (3.2.5c)$$

where

$$\Psi_i(k) \triangleq (\Gamma_i(k))^{-1} B_i^T K_i(k+1), \; i=1.2 . \qquad (3.2.6)$$

$$\Xi_1(k) \triangleq I - \Psi_1(k) B_2 \Psi_2(k) B_1 . \text{ and} \qquad (3.2.7a)$$

$$\Xi_2(k) \triangleq I - \Psi_2(k) B_1 \Psi_1(k) B_2 . \qquad (3.2.7b)$$

3.2.2 Iterating a Riccati Equation

This subsection determines an algorithm for iterating a Riccati equation assuming the feedbacks are known. The goal here is to minimize the number of inverses appearing in equation (3.2.1). From a computational viewpoint, the right-hand side of the Riccati equation (3.2.1) is most efficiently calculated by using a slightly modified form of (3.2.1). To see this, start with (3.2.1) and use the relations (3.2.4) and (3.2.5) to write

$$E^T K_i(k) E$$

$$= A_i^T K_i(k+1) A_i - A_i^T K_i(k+1) B_i (\Gamma_i)^{-1} B_i^T K_i(k+1) A_i + Q_i \tag{3.2.8a}$$

$$= A_i^T K_i(k+1) A_i - A_i^T K_i(k+1) B_i (\Gamma_i)^{-1} \left[\Gamma_i \Gamma_i^{-1}\right] B_i^T K_i(k+1) A_i + Q_i \tag{3.2.8b}$$

$$= A_i^T K_i(k+1) A_i - (A_i^T K_i(k+1) B_i \Gamma_i^{-1}) \Gamma_i (\Gamma_i^{-1} B_i^T K_i(k+1) A_i) + Q_i \tag{3.2.8c}$$

$$= A_i^T K_i(k+1) A_i - F_i^T \Gamma_i F_i + Q_i \tag{3.2.8d}$$

where the dependence on k has been dropped for all variables except K_i.

Remark : It is a simple exercise to verify that (3.2.8d) can be written as:

$$E^T K_i(k) E = A_{CL}^T K_i(k+1) A_{CL} + F_i^T R_i F_i + Q_i \tag{3.2.8e}$$

where $A_{CL} \stackrel{\Delta}{=} A - B_1 F_1 - B_2 F_2$. This form is as compact as (3.2.8d). Indeed, this computationally attractive form may be found in [7,p.253] where the setting is largely theoretical. However, it will become evident that it is less efficient to use (3.2.8e) to iterate the Riccati equations, because the Cholesky factor of R_i is never computed whereas the Cholesky factor of Γ_i will be available from another calculation.

One additional simplification is possible. In view of (3.2.6), equation (3.2.8c) can be rewritten as:

$$E^T K_i(k) E = A_i^T \left[K_i(k+1) - \Psi_i^T \Gamma_i \Psi_i\right] A_i + Q_i \tag{3.2.8f}$$

which is extremely compact.

It is true that each Riccati equation can be computed directly as a function of $K_1(k)$ and $K_2(k)$, but the resulting equation is very complex and quite often the feedbacks are needed.

Furthermore, once the feedbacks are calculated, the Riccati iterations are most efficiently computed using (3.2.8f) - *an equation with no inverses*.

Before presenting the algorithm for iterating a Riccati equation, several small algorithms must be defined. This is done in the next few subsections. Algorithm QDFORM computes a quadratic form or variations of a quadratic form. Algorithm PSICOM computes the variable $\Psi_i(k)$ as defined in (3.2.6). Algorithm XTRACT computes $K_i(k)$ given the right-hand side of (3.2.8f). Algorithm DISRIC combines these algorithms to perform one iteration of a Riccati equation.

3.2.2.1 Algorithm QDFORM

It is clear that the quadratic form $B^T A B$ where B is arbitrary and A is a positive-(semi)definite, symmetric matrix appears frequently. Anticipating the need to compute several quadratic forms, let us introduce the following algorithm.

Algorithm QDFORM

Step 1 : Factor $A \triangleq X_A^T X_A$ where X_A is upper triangular. X_A is called the Cholesky factor of A and is obtained via the LINPACK subroutine SPOFA.

Step 2 : Compute $Y = X_A B$ taking advantage of the fact that X_A is upper triangular.

Step 3 : Compute $D = Y^T Y$ taking advantage of the fact that D is symmetric.

Remarks : As coded in Appendix A. Step 1 can be bypassed if the Cholesky factor of A is already available. Steps 2 and 3 require one optimized matrix multiply DO-Loop in FORTRAN. Also, if the quantity $C + B^T A B$, where C is symmetric, is desired, then algorithm QDFORM is applicable if Step 3 is modified to include the array C as the initial condition in the multiply DO-Loop. The same can be said for the form $C - B^T A B$.

3.2.2.2 Algorithm PSICOM

The computation of $\Psi_i(k)$, i=1,2 establishes the foundation upon which all subsequent calculations are built. Observe that (3.2.4), (3.2.5), (3.2.7) and (3.2.8) depend on $\Psi_i(k)$. It is apparent that a thorough investigation of the computation of $\Psi_i(k)$ is needed. From the definition (3.2.6), it would seem that the explicit computation of $(\Gamma_i(k))^{-1}$ is necessary. However, this is not the case. Since $\Gamma_i(k)$ is a positive-definite, symmetric matrix, the LINPACK [24, Chapter 3] software for general positive-definite matrices is used to circumvent this problem.

Although $\Psi_i(k)$ can be found in 4 easy steps, the result of each step of the calculation is needed in later computations. For example, the quantity W defined in Step 1 of algorithm PSICOM is used later, so let

$$W_1(k) \stackrel{\Delta}{=} B_1^T K_1(k+1), \text{ and} \tag{3.2.9a}$$

$$W_2(k) \stackrel{\Delta}{=} B_2^T K_2(k+1) . \tag{3.2.9b}$$

Hence it is desirable to create an algorithm that computes and returns $\Psi_i(k)$ as well as all the intermediate quantities. Since $\Psi_i(k)$ is a function of Γ_i, B_i and K_i, the subscript i can be dropped. Also, the dependency on k can be suppressed. The algorithm for computing Ψ is now stated.

Algorithm PSICOM

Step 1 : Compute $W \stackrel{\Delta}{=} B^T K$ using algorithm MLTPLY.

Step 2 : Compute Γ using algorithm QDFORM.

Step 3 : Factor $\Gamma = X_\Gamma^T X_\Gamma$ where X_Γ is upper triangular. X_Γ is called the Cholesky factor of Γ and is obtained using the LINPACK subroutine SPOFA.

Step 4 : Form $\Psi = \Gamma^{-1} W$ without explicitly computing Γ^{-1} using the LINPACK subroutine SPOSL.

Remarks : Step 4 solves the linear equation $\Gamma \Psi = W$ for Ψ one column at a time using the Cholesky factor X_Γ. *Thus, it is not necessary to explicitly form the inverse of* Γ. Algorithm

PSICOM takes B, K, and R as input and subsequently produces W, Γ, X_Γ, and Ψ as output. Hence, if Ψ is needed for a later calculation and algorithm PSICOM is invoked, then Γ need not be computed using algorithm QDFORM because PSICOM will return it as a byproduct of the calculation of Ψ. The Riccati algorithms will exploit this savings in computation.

3.2.2.3 Algorithm XTRACT

Given the right-hand side of (3.2.8f), the quantity $K_i(k)$ is immediately known only if $E = \mathbf{I}$. If E differs from the identity matrix, then additional steps must be taken to extract $K_i(k)$ from the quadratic form $E^T K_i(k) E$. Algorithm XTRACT performs exactly that function. Specifically, suppose the right-hand side of (3.2.8f) evaluates to a positive-definite, symmetric matrix called Z. Then Z has a Cholesky factorization, $Z \triangleq X_Z^T X_Z$. Similarly, the kernel of the left-hand side of (3.2.8f) has a Cholesky factorization $K \triangleq X_K^T X_K$. Equating both quantities yields

$$E^T X_K^T X_K E = X_Z^T X_Z \ .$$

Algorithm XTRACT is based on this observation.

Since E is assumed to be dense and with no special structure, the LINPACK software for general matrices is employed. Assumption 2.1 guarantees that the inverse of E exists. However, as with algorithm PSICOM, the inverse is never explicitly computed. Instead the LU factorization of E is used. Now, the algorithm for extracting $K_i(k)$ is stated.

Algorithm XTRACT

Step 1: Factor $Z = X_Z^T X_Z$ where X_Z is upper triangular. X_Z is called the Cholesky factor of Z and is obtained using the LINPACK subroutine SPOFA.

Step 2: Factor $E = L_E U_E$ where U_E is upper triangular and L_E is the product of elementary lower triangular and permutation matrices. $L_E U_E$ is called the LU factorization of E and is obtained using the LINPACK subroutine SGEFA.

Step 3 : Solve the linear system $E^T X_K^T = X_Z^T$ for X_K without explicitly computing E^{-1} using the LINPACK subroutine SGESL.

Step 4 : Compute $K = X_K^T X_K$ taking advantage of the fact that D is symmetric.

Remarks : Step 3 solves the linear equation $E^T X_K^T = X_Z^T$ for X_K one column at a time using the LU factorization $L_E U_E$. *Thus, it is not necessary to explicitly form the inverse of* E. Step 4 requires one optimized matrix multiply DO-Loop in FORTRAN. Algorithm XTRACT is only invoked if $E \neq \mathbf{I}$. However, if E should possess additional structure (e.g., diagonal), then steps 2 and 3 should be replaced by another algorithm which exploits that structure.

3.2.2.4 Algorithm DISRIC

This algorithm combines the previous algorithms to perform one iteration of a Riccati equation. It is assumed that the feedbacks (and therefore A_i, $i \in \mathbf{N}$) are known. First the right-hand side of (3.2.8f) is computed. Then $K_i(k)$ is extracted. The algorithm for iterating a Riccati equation is stated below.

Algorithm DISRIC

Step 1 : Compute Ψ using algorithm PSICOM. Γ and X_Γ will also be computed and provided upon return.

Step 2 : Form $V = K - \Psi^T \Gamma \Psi$ using algorithm QDFORM taking advantage of the fact that X_Γ is already available.

Step 3 : Form $Z = Q + A^T V A$ using algorithm QDFORM.

Step 4 : Compute $K(k)$ from the right-hand side of (3.2.8f) using algorithm XTRACT, if necessary.

Remarks : Notice that every step of algorithm DISRIC is a call to one of the previously-defined algorithms. This indicates that the low-level routines are very modular. This is one of the

trademarks of a good structured-programming approach.

3.2.3 Calculating the Feedbacks

Having devised a method for iterating a Riccati equation, we turn our attention to calculating the feedbacks. Using (3.2.6) we can rewrite (3.2.4a) and (3.2.5a) as

$$F_i(k) = \Psi_i(k) A_i(k), \; i \in \mathbf{N}. \tag{3.2.10}$$

But, this form assumes knowledge of the other DM's feedback matrix. Initially neither F_1 nor F_2 is known. Hence either (3.2.4c) or (3.2.5c) must be used to compute one of these matrices based upon some criterion. Each equation is undesirable because it contains an inverse. Thus, it is ultimately necessary to compute the inverse of one general square matrix. Furthermore, the decoupling procedure used to produce (3.2.4c) and (3.2.5c) clouds the presence of symmetric and positive-definite matrices. Therefore, it is worthwhile to study these equations with the intention of exposing any additional symmetry and/or positive definiteness. Towards this end take (3.2.4c) and premultiply by $\Xi_1(k)$. The result is

$$\Xi_1(k) F_1(k) = \Psi_1(k) \left[I - B_2 \Psi_2(k) \right] A \; . \tag{3.2.11a}$$

Substitute (3.2.6) and (3.2.7a) into (3.2.11a) to obtain

$$\left[I - (\Gamma_1(k))^{-1} B_1^T K_1(k+1) B_2 (\Gamma_2(k))^{-1} B_2^T K_2(k+1) B_1 \right] F_1(k) \tag{3.2.11b}$$

$$= (\Gamma_1(k))^{-1} B_1^T K_1(k+1) A - (\Gamma_1(k))^{-1} B_1^T K_1(k+1) B_2 (\Gamma_2(k))^{-1} B_2^T K_2(k+1) A \; .$$

Premultiplying by $\Gamma_1(k)$ yields

$$\left[\Gamma_1(k) - B_1^T K_1(k+1) B_2 (\Gamma_2(k))^{-1} B_2^T K_2(k+1) B_1 \right] F_1(k) \tag{3.2.11c}$$

$$= B_1^T K_1(k+1) A - B_1^T K_1(k+1) B_2 (\Gamma_2(k))^{-1} B_2^T K_2(k+1) A \; .$$

Define the positive-(semi)definite, symmetric matrices

$$\underline{\chi}_i(k) \stackrel{\Delta}{=} B_i (\Gamma_i(k))^{-1} B_i^T \; , \; i \in \mathbf{N} \; . \tag{3.2.12}$$

Then in view of (3.2.9) and (3.2.12), equation (3.2.11c) reduces to

$$\left[\Gamma_1(k) - W_1(k) \underline{\chi}_2(k) K_2(k+1) B_1 \right] F_1(k) = W_1(k) \left[I - \underline{\chi}_2(k) K_2(k+1) \right] A \; . \tag{3.2.13a}$$

A similar derivation applied to (3.2.5c) gives

$$\left[\Gamma_2(k) - W_2(k)\,\chi_1(k)\,K_1(k+1)\,B_2\right] F_2(k) = W_2(k)\left[I - \chi_1(k)\,K_1(k+1)\right] A \quad . \quad (3.2.13b)$$

Define

$$\Upsilon_1(k) \triangleq \Gamma_1(k) - W_1(k)\,\chi_2(k)\,K_2(k+1)\,B_1, \text{ and} \quad (3.2.14a)$$

$$\Upsilon_2(k) \triangleq \Gamma_2(k) - W_2(k)\,\chi_1(k)\,K_1(k+1)\,B_2 \quad . \quad (3.2.14b)$$

Then, it is wise to choose $\Upsilon_i(k)$ such that it has smallest condition number over all other Υ_i's, $i \in N$. Specifically, suppose that DM $\hat{i}$ has $\Upsilon_{\hat{i}}(k)$ with smallest condition number. Obviously we want to compute $\left[\Upsilon_{\hat{i}}\right]^{-1}$ first.

It is apparent that the matrices $\chi_i(k)$, $i \in N$ are needed for the feedback calculation. Since algorithm QDFORM is not suited to handle an inverse as the kernel of the quadratic form, algorithm CHICOM is presented.

Algorithm CHICOM

Step 1: Compute $Y = \left[X_\Gamma\right]^{-T} B^T$ using LINPACK subroutine STRSL. X_Γ is the Cholesky factor of Γ obtained from algorithm PSICOM.

Step 2: Compute $\chi = Y^T Y$ taking advantage of the fact that χ is symmetric.

The following algorithm determines $\hat{i}$ and also computes $\Upsilon_i(k)$, for all $i \in N$. Since the final result of this algorithm is the computation of $\hat{i}$, it is most advantageous to code this as an INTEGER FUNCTION in FORTRAN which returns an integer equal to $\hat{i}$. Note that for this algorithm the dependence on $i \in N$ cannot be suppressed.

Algorithm EYEHAT

Step 1: Compute $\chi_i(k)$, $i \in N$ using algorithm CHICOM.

Step 2: Compute $T_1(k) = W_1(k)\,\chi_2(k)$ and $T_2(k) = W_2(k)\,\chi_1(k)$ using algorithm MLTPLY.

Step 3 : Compute $V_1(k) = T_1(k)\, K_2(k+1)$ and $V_2(k) = T_2(k)\, K_1(k+1)$ using algorithm MLTPLY.

Step 4 : Compute $\Upsilon_1(k) = \Gamma_1(k) - V_1(k)\, B_1$ and $\Upsilon_2(k) = \Gamma_2(k) - V_2(k)\, B_2$ in one matrix multiply DO-Loop.

Step 5 : Factor $\Upsilon_1(k) = L_{\Upsilon_1} U_{\Upsilon_1}$ and $\Upsilon_2(k) = L_{\Upsilon_2} U_{\Upsilon_2}$ where U_{Υ_i} is an upper triangular matrix and L_{Υ_i} is the product of elementary lower triangular and permutation matrices, $i \in N$. These factors are obtained via the LINPACK subroutine SGECO. As a byproduct of this subroutine call, an estimate of the conditions numbers of Υ_i, $\kappa_i \underline{\Delta} \kappa(\Upsilon_i)$, is returned.

Step 6 : Compute $\hat{i}$ such that $\kappa_{\hat{i}} = \min_{i \in N} \{\kappa_i\}$.

Remarks : Since $\Upsilon_i(k)$ has no special structure and an estimate of the condition number is needed, the LINPACK subroutine SGECO is the obvious choice in step 5. Algorithm EYEHAT is computationally intensive! This is the price paid for basing the choice of which matrix to invert on the lowest condition number criterion. Another guideline might be to invert the matrix with smallest dimension. This would lessen the computational burden born by the algorithm, but might lead to inaccurate results in unusually conditioned circumstances.

It is clear that the first feedback must be found by finding an inverse of an arbitrary matrix. Subsequently, the other feedback can be calculated via (3.2.10) using algorithm MLTPLY. The next algorithm computes the feedbacks for a 2-player game where the value of $\hat{i}$ is already available from the criterion defined in algorithm EYEHAT.

Algorithm FEEDBK

Step 1 : Compute $T_i = \left[\Upsilon_i\right]^{-1} W_i$ using the LINPACK subroutine SGESL where the L-U decomposition is obtained from algorithm EYEHAT.

Step 2 : Compute $U_j = I - X_j(k)\, K_j(k+1)$, $j \neq \hat{i}$ in one matrix multiply Do-Loop.

Step 3 : Compute $V_j = T_j\, U_j$ using algorithm MLTPLY.

Step 4 : Compute $F_{\hat{i}}(k) = V_{\hat{i}} A$ using algorithm MLTPLY.

Step 5 : Form $A_j = A - B_{\hat{i}} F_{\hat{i}}$ where $j \neq \hat{i}$ in one matrix multiply DO-Loop.

Step 6 : Form the other feedback $F_j = \Psi_j A_j$ where $j \neq \hat{i}$.

Remarks : Algorithm FEEDBK is sufficient for computing the feedbacks only. However, the ultimate goal is to perform one iteration of each Riccati equation described by (3.2.8). For that reason, it is desirable to modify step 5 above so that other quantities are available. Specifically, change it to Form $A_1 = A - B_2 F_2$ and $A_2 = A - B_1 F_1$ in one matrix multiply DO-Loop.

3.2.4 Computation of Coupled Riccati Iterations

From (3.2.8f), notice that if $F_i = \mathbf{0}$, then just compute the Lyapunov equation

$$E^T K_i(k) E = A_i^T(k) K_i(k+1) A_i(k) + Q_i \tag{3.2.15}$$

using algorithm QDFORM. Otherwise, if $F_j = \mathbf{0}$, $j \neq i$ then compute

$$E^T K_i(k) E = A^T \left[K_i(k+1) - \Psi_i^T \Gamma_i \Psi_i \right] A + Q_i \tag{3.2.16}$$

using algorithm DISRIC. Regardless of the condition encountered, the following algorithm performs one iteration of coupled discrete-time Riccati equations.

Algorithm RICCAT

Step 1 : Compute the feedbacks using algorithm FEEDBK.

Step 2 : Iterate each Riccati equation using algorithm DISRIC.

Remarks : Computation of Multirate Coupled Riccati Iterations

If $L \in \mathbf{Z}^+$, the Riccati equations to be iterated are the same. However, whenever $L \notin \mathbf{Z}^+$, several simplifications are possible. First,

$$F_1(k) = \Psi_1(k) A \tag{3.2.17a}$$

$$F_2(k) = \mathbf{0} \ . \tag{3.2.17b}$$

Hence, the Riccati equations that have to be iterated reduce to

$$E^T K_1(k) E = A^T \left[K_1(k+1) - \Psi_1^T \Gamma_1 \Psi_1 \right] A + Q_1 \tag{3.2.18a}$$

$$E^T K_2(k) E = A_2^T(k) K_2(k+1) A_2(k) + Q_2 \ . \tag{3.2.18b}$$

Concluding Remarks :

Section 3.2 describes the numerical solution of a coupled discrete-time Riccati equation problem motivated in Chapters 1 and 2. During the development of the solution method, several algorithms are defined. Collectively, they serve to provide a rich numerical foundation upon which more sophisticated algorithms can be built. Thus, the Riccati algorithms presented here are an example of using software to efficiently solve a frequently-formulated game problem. This is the overall spirit behind CACSD endeavors. The results discussed in this subsection apply to the standard LQ regulator problem and the Nash equilibrium solution of an LQ state-feedback problem (the coupled Riccati case). The issues associated with solving single and coupled discrete-time Riccati equations are delineated. As a consequence of this analysis, it is shown that the numerical intensiveness is directly related to the criterion used for determining $\hat{i}$ via algorithm EYEHAT. Beyond that fact, the Riccati iteration procedure basically boils down to computing one or more quadratic forms, most conveniently calculated by algorithm QDFORM.

3.3 Finite Horizon Problems — Existence Issues

In this section we investigate conditions for which the existence of solutions to finite-horizon problems is guaranteed. To begin with, Proposition 3.1 in Section 3.1 is based upon a singular value argument applied to $\Xi_i(k)$. However utilizing the definition (3.1.7), a stronger result can be stated. Define :

$$\Omega_{12}(k) \underline{\Delta} \Psi_1(k) B_2 \tag{3.3.1a}$$

$$\Omega_{21}(k) \underline{\Delta} \Psi_2(k) B_1 \ . \tag{3.3.1b}$$

Then, $\Xi_1(k) = I - \Omega_{12}(k)\ \Omega_{21}(k)$ and $\Xi_2(k) = I - \Omega_{21}(k)\ \Omega_{12}(k)$. Hence,

Proposition **3.2 :** Existence of Feedbacks for Finite-Time Problems

$K_1(k)$ exists if and only if $\lambda(\ \Omega_{12}(k)\ \Omega_{21}(k)) \neq 1$ for all eigenvalues. $K_2(k)$ exists if and only if $\lambda(\ \Omega_{21}(k)\ \Omega_{12}(k)) \neq 1$ for all eigenvalues.

Proof : Apply the similarity transform that puts $\Omega_{12}(k)\ \Omega_{21}(k)$ into Jordan form to $\Xi_1(k)$ and the first statement follows immediately. Apply the similarity transform that puts $\Omega_{21}(k)\ \Omega_{12}(k)$ into Jordan form to $\Xi_1(k)$ and the second statement follows immediately.

□

3.4 Infinite Horizon Problems – Convergence Issues

Before concluding this chapter, an investigation of the existence of solutions to the infinite-time LQ Nash game is conducted. To the author's knowledge, virtually no work has produced either necessary or sufficient conditions regarding existence of solutions even though the theory governing the one-player optimal infinite-time LQ regulator problem is well-established [5,48]. Solutions to single-rate infinite-time LQ Nash games will be studied. We determine conditions under which the existence of an infinite-time solution is guaranteed.

3.4.1 Preliminaries

Let $\boldsymbol{X}$ denote the space of all nxn symmetric matrices. Let $\boldsymbol{Y} \subset \boldsymbol{X}$ denote the set of all positive-semidefinite symmetric matrices in $\boldsymbol{X}$. Clearly $\boldsymbol{X}$ is a linear vector space and $\boldsymbol{Y}$ is a closed subset of $\boldsymbol{X}$. Let $X, Y \in \boldsymbol{X}$ be any two arbitrary elements of the space $\boldsymbol{X}$. Then

$$(X, Y) \triangleq \begin{vmatrix} X & 0 \\ 0 & Y \end{vmatrix}$$

denotes an arbitrary element of the product space $\boldsymbol{X} \times \boldsymbol{X}$. Now, let $\| \cdot \|_2$ denote the standard induced matrix 2-norm. Then for any $(X, Y) \in \boldsymbol{X} \times \boldsymbol{X}$, define

$$\| (X, Y) \| \triangleq \| X \|_2 + \| Y \|_2 \tag{3.4.1}$$

to be the norm on the product space. Obviously, the space $\boldsymbol{X} \times \boldsymbol{X}$ with the norm defined by (3.4.1)

is a complete normed linear vector space. Hence, it is a Banach space.

Assumption **3.1 :** Throughout the remainder of this section, the matrix E is always assumed to equal the identity.

We have two maps :

$$\boldsymbol{R}_1 : \boldsymbol{X} \times \boldsymbol{X} \rightarrow \boldsymbol{X} \quad \text{and} \quad \boldsymbol{R}_2 : \boldsymbol{X} \times \boldsymbol{X} \rightarrow \boldsymbol{X}.$$

Specifically, from (2.1.11c), consider

$$\boldsymbol{R}_1(X,Y) \triangleq A_1^T(X,Y)\left[X^{-1} + B_1 R_1^{-1} B_1^T\right]^{-1} A_1(X,Y) + Q_1$$

$$\boldsymbol{R}_2(X,Y) \triangleq A_2^T(X,Y)\left[Y^{-1} + B_2 R_2^{-1} B_2^T\right]^{-1} A_2(X,Y) + Q_2$$

where $A_1(X,Y) \triangleq A - B_2 F_2(X,Y)$ and $A_2(X,Y) \triangleq A - B_1 F_1(X,Y)$. Define

$$\Phi_i(X) \triangleq \left[X^{-1} + B_i R_i^{-1} B_i^T\right]^{-1}.$$

Then,

$$\left.\begin{aligned} \boldsymbol{R}_1(X,Y) &\triangleq A_1^T(X,Y)\,\Phi_1(X)\,A_1(X,Y) + Q_1 \\ \boldsymbol{R}_2(X,Y) &\triangleq A_2^T(X,Y)\,\Phi_2(Y)\,A_2(X,Y) + Q_2 \end{aligned}\right\}. \tag{3.4.2}$$

Notice that $A_1(X,Y)$ and $A_2(X,Y)$ serve to *couple* the two equations of (3.4.2). Consider stacking the previous two equations. Then define :

$$\mathbf{R}(X,Y) \triangleq \begin{bmatrix} \boldsymbol{R}_1(X,Y) & 0 \\ 0 & \boldsymbol{R}_2(X,Y) \end{bmatrix}.$$

Given this setup we observe that

$$\mathbf{R} : \boldsymbol{X} \times \boldsymbol{X} \rightarrow \boldsymbol{X} \times \boldsymbol{X}.$$

The following result is the most important one of this chapter. It provides sufficient conditions for existence of and convergence to the single-rate infinite-time LQ Nash equilibrium solution.

Theorem **3.1 :** Existence and Convergence of Solutions to the Infinite-Time LQ Nash Game

Given the system (2.1.1)-(2.1.2) with performance indices (2.1.3), then there exist constants, $0 < \bar{\epsilon_1} < 1$ and $0 < \bar{\epsilon_2} < 1$ which define the matrices $\bar{R}_1 \triangleq \frac{1}{\bar{\epsilon_1}} I$ and $\bar{R}_2 \triangleq \frac{1}{\bar{\epsilon_2}} I$ such that for all R_1 and R_2 where $\underline{\sigma}(R_1) > \frac{1}{\bar{\epsilon_1}}$ and $\underline{\sigma}(R_2) > \frac{1}{\bar{\epsilon_2}}$ and all system matrices A with $\| A \| < \bar{\xi}(\bar{\epsilon_1}, \bar{\epsilon_2}) < 1$, the set of coupled difference equations :

$$\left.\begin{aligned} K_1(k) &= A_1^T(k) \left| (K_1(k+1))^{-1} + B_1(R_1(k))^{-1} B_1^T \right|^{-1} A_1(k) + Q_1 \\ K_2(k) &= A_2^T(k) \left| (K_2(k+1))^{-1} + B_2(R_2(k))^{-1} B_2^T \right|^{-1} A_2(k) + Q_2 \end{aligned}\right\} \tag{3.4.3}$$

constitutes a contraction mapping and hence converges to a unique fixed point denoted (K_1^*, K_2^*).

Proof : To begin with, let $\bar{\epsilon_1} \to 0$ and $\bar{\epsilon_2} \to 0$ independently. From Theorem 2.1 we know that in the limit, $\| F_1 \| = \| F_2 \| = 0$ which implies that $A_1 \to A$ and $A_2 \to A$ in the limit. Furthermore, $\|(\bar{R}_1)^{-1}\| = \bar{\epsilon_1} \to 0$ and $\|(\bar{R}_2)^{-1}\| = \bar{\epsilon_2} \to 0$ by construction. Hence, the two *coupled* discrete-time Riccati equations tend toward two *decoupled* discrete-time Lyapunov equations given by

$$\left.\begin{aligned} K_1(k) &= A^T K_1(k+1) A + Q_1 \\ K_2(k) &= A^T K_2(k+1) A + Q_2 \end{aligned}\right\} \tag{3.4.4}$$

in the limit as $\bar{\epsilon_1} \to 0$ and $\bar{\epsilon_2} \to 0$.

It is well known that this set of equations has a unique positive-definite symmetric fixed point (K_1^*, K_2^*) if and only if $\| A \| < 1$. Moreover, (3.4.4) is a contraction mapping, so that for any positive-semidefinite symmetric initial guess $(K_1(0), K_2(0))$, convergence to the fixed point is assured as $k \to -\infty$.

Now we must argue that for a given problem, there exists an open neighborhood of $\bar{R}_1^{-1} = \bar{R}_2^{-1} = \mathbf{0}$ characterized by the constants $\bar{\epsilon_1}$ and $\bar{\epsilon_2}$ such that for all A with $\| A \| < \bar{\xi}(\bar{\epsilon_1}, \bar{\epsilon_2}) < 1$, equation (3.4.3) constitutes a contraction mapping. The next subsection will more than fulfill this requirement.

□

3.4.2 Contraction Mapping Argument

If the mapping $\mathbf{R}$ is to be a contraction, then the following condition must be satisfied :

$$\| \mathbf{R}(X,Y) - \mathbf{R}(Z,W) \| \leq \alpha \| (X,Y) - (Z,W) \| \tag{3.4.5}$$

for all $X, Y, Z, W \in \Upsilon$ and $0 \leq \alpha < 1$. Straightforwardly, compute :

$$\| \mathbf{R}(X,Y) - \mathbf{R}(Z,W) \| = \| R_1(X,Y) - R_1(Z,W) \|_2 + \| R_2(X,Y) - R_2(Z,W) \|_2 .$$

Similarly,

$$\| (X,Y) - (Z,W) \| = \| X - Z \|_2 + \| Y - W \|_2 .$$

Therefore, the condition (3.4.5) is equivalent to

$$\left. \begin{array}{l} \| R_1(X,Y) - R_1(Z,W) \|_2 + \| R_2(X,Y) - R_2(Z,W) \|_2 \\ \quad \leq \alpha \left(\| X - Z \|_2 + \| Y - W \|_2 \right) \end{array} \right\} . \tag{3.4.6}$$

The purpose of the ensuing discussion is to completely characterize those cases for which (3.4.6) holds. The contraction mapping that we seek occurs on a closed subset of $\Upsilon \times \Upsilon$. Hence it must be shown that there exists a region $\Delta \subset \Upsilon \times \Upsilon$ where (3.4.2) maps Δ into itself. Toward this end, the following facts are established.

Fact **3.1** : Given any $X \in \Upsilon$, then $\| \Phi_i(X) \|_2 \leq \| X \|_2$ for all $i \in \{ 1,2 \}$.

Proof : Since $X \geq 0$ and $B_i R_i^{-1} B_i^T \geq 0$,

$$\| \Phi_i(X) \|_2 = \frac{1}{\underline{\sigma}(X^{-1} + B_i R_i^{-1} B_i^T)} \leq \frac{1}{\underline{\sigma}(X^{-1})} = \| X \|_2$$

where $\underline{\sigma}(\cdot)$ denotes the smallest singular value of $(\cdot)$.

□

To simplify the derivation, introduce the following definitions :

$$\rho_i(X,Y) \stackrel{\Delta}{=} \| A_i(X,Y) \|_2 \quad , i=1,2 \tag{3.4.7}$$

$$\delta_i(\rho_i) \stackrel{\Delta}{=} \frac{\| Q_i \|_2}{1 - (\rho_i(X,Y))^2} \quad , i=1,2 \tag{3.4.8}$$

$$B_\delta \triangleq \{ X \in \mathcal{X} \mid \| X \|_2 \leq \delta \} . \tag{3.4.9}$$

Equation (3.4.9) describes a closed ball of radius δ.

Then we have :

Claim **3.1** : Existence of a Region where the Riccati maps are Into

Suppose $\rho_i(X, Y) \leq \overline{\rho}_i < 1$ and $\overline{\delta}_i \triangleq \dfrac{\| Q_i \|_2}{1-(\overline{\rho}_i)^2}$ for i=1,2. Given any $(X, Y) \in \mathcal{Y} \times \mathcal{Y}$ such that $\| X \|_2 \leq \overline{\delta}_1$ and $\| Y \|_2 \leq \overline{\delta}_2$, then for all such X, Y it follows that $(\tilde{X}, \tilde{Y}) \triangleq \mathbf{R}(X, Y)$ has the property that $\| \tilde{X} \|_2 \leq \overline{\delta}_1$ and $\| \tilde{Y} \|_2 \leq \overline{\delta}_2$. Hence, $\mathbf{R}(X, Y)$ maps $\Delta \triangleq B_{\overline{\delta}_1} \times B_{\overline{\delta}_2} \subset \mathcal{Y} \times \mathcal{Y}$ into itself.

Proof : Notice that from (3.4.2) :

$$\begin{aligned} \| \tilde{X} \|_2 &= \| A_1^T(X, Y) \Phi_1(X) A_1(X, Y) + Q_1 \|_2 \\ &\leq \| A_1(X, Y) \|_2^2 \cdot \| \Phi_1(X) \|_2 + \| Q_1 \|_2 \\ &\leq \| A_1(X, Y) \|_2^2 \cdot \| X \|_2 + \| Q_1 \|_2 \end{aligned}$$

where Fact 3.1 has been utilized in the last step.

Thus,

$$\begin{aligned} \| \tilde{X} \|_2 &\leq (\rho_1(X, Y))^2 \cdot \| X \|_2 + \| Q_1 \|_2 \\ &\leq (\overline{\rho}_1)^2 \cdot \overline{\delta}_1 + \| Q_1 \|_2 = \frac{(\overline{\rho}_1)^2 \| Q_1 \|_2}{1-(\overline{\rho}_1)^2} + \| Q_1 \|_2 = \overline{\delta}_1 . \end{aligned}$$

Similarly,

$$\begin{aligned} \| \tilde{Y} \|_2 &= \| A_2^T(X, Y) \Phi_2(Y) A_2(X, Y) + Q_2 \|_2 \\ &\leq \| A_2(X, Y) \|_2^2 \cdot \| Y \|_2 + \| Q_2 \|_2 \\ &\leq (\overline{\rho}_2)^2 \cdot \overline{\delta}_2 + \| Q_2 \|_2 \\ &= \frac{(\overline{\rho}_2)^2 \| Q_2 \|_2}{1-(\overline{\rho}_2)^2} + \| Q_2 \|_2 = \overline{\delta}_2 . \end{aligned}$$

Therefore, $\| \tilde{X} \|_2 \leq \overline{\delta}_1$, $\| \tilde{Y} \|_2 \leq \overline{\delta}_2$, and $(\tilde{X}, \tilde{Y}) \in \Delta \triangleq B_{\overline{\delta}_1} \times B_{\overline{\delta}_2}$.

□

Now, we proceed to derive conditions such that (3.4.6) holds in the region Δ. In particular, focus on the map $R_1(X,Y)$. A parallel argument can be developed for the map $R_2(X,Y)$. Given any (X,Y) and $(Z,W) \in \mathcal{X} \times \mathcal{X}$,

$$\left.\begin{aligned} & R_1(X,Y) - R_1(Z,W) \\ & \quad = A_1^T(X,Y)\,\Phi_1(X)\,A_1(X,Y) - A_1^T(Z,W)\,\Phi_1(Z)\,A_1(Z,W) \end{aligned}\right\} . \tag{3.4.10}$$

Application of the matrix identity

$$WX - YZ = \frac{1}{2} \cdot \left| (W-Y)(X+Z) + (W+Y)(X-Z) \right| \tag{3.4.11}$$

to (3.4.10) yields :

$$\begin{aligned} & R_1(X,Y) - R_1(Z,W) \\ & \quad = \frac{1}{2} \cdot \left| \left| A_1^T(X,Y) - A_1^T(Z,W) \right| \cdot \left| \Phi_1(X)\,A_1(X,Y) + \Phi_1(Z)\,A_1(Z,W) \right| \right. \\ & \qquad \left. + \left| A_1^T(X,Y) + A_1^T(Z,W) \right| \cdot \left| \Phi_1(X)\,A_1(X,Y) - \Phi_1(Z)\,A_1(Z,W) \right| \right| \\ & \quad = \frac{1}{2} \cdot \left| \left| \left(B_2 \left(F_2(Z,W) - F_2(X,Y) \right) \right)^T \cdot \left| \Phi_1(X)\,A_1(X,Y) + \Phi_1(Z)\,A_1(Z,W) \right| \right. \right. \\ & \qquad \left. \left. + \left| A_1^T(X,Y) + A_1^T(Z,W) \right| \cdot \left| \Phi_1(X)\,A_1(X,Y) - \Phi_1(Z)\,A_1(Z,W) \right| \right| . \end{aligned}$$

But, from an additional use of (3.4.11)

$$\begin{aligned} & \Phi_1(X)\,A_1(X,Y) - \Phi_1(Z)\,A_1(Z,W) \\ & \quad = \frac{1}{2} \cdot \left| \left| \Phi_1(X) - \Phi_1(Z) \right| \cdot \left| A_1(X,Y) + A_1(Z,W) \right| \right. \\ & \qquad \left. + \left| \Phi_1(X) + \Phi_1(Z) \right| \cdot \left| A_1(X,Y) - A_1(Z,W) \right| \right| . \end{aligned}$$

Therefore, we conclude that :

$$\begin{aligned} & R_1(X,Y) - R_1(Z,W) \\ & \quad = \frac{\left| B_2 \left(F_2(Z,W) - F_2(X,Y) \right) \right|^T}{2} \cdot \left| \Phi_1(X)\,A_1(X,Y) + \Phi_1(Z)\,A_1(Z,W) \right| \end{aligned}$$

$$+ \frac{\left| A_1^T(X,Y) + A_1^T(Z,W) \right|}{4} \cdot \left| \left| \Phi_1(X) - \Phi_1(Z) \right| \left| A_1(X,Y) + A_1(Z,W) \right| \right|$$

$$+ \frac{\left| A_1^T(X,Y) + A_1^T(Z,W) \right|}{4} \cdot \left| \left| \Phi_1(X) + \Phi_1(Z) \right| \left| B_2\left(F_2(Z,W) - F_2(X,Y) \right) \right| \right| .$$

Applying the matrix 2-norm and then the triangle inequality on the last equation gives

$$\left.\begin{aligned} &\| R_1(X,Y) - R_1(Z,W) \|_2 \\ &\leqslant \frac{\| \Phi_1(X) A_1(X,Y) + \Phi_1(Z) A_1(Z,W) \|_2}{2} \cdot \| B_2\left(F_2(Z,W) - F_2(X,Y) \right) \|_2 + \\ &\frac{\| A_1(X,Y) + A_1(Z,W) \|_2^2}{4} \cdot \| \Phi_1(X) - \Phi_1(Z) \|_2 + \\ &\frac{\| A_1(X,Y) + A_1(Z,W) \|_2}{4} \cdot \| \Phi_1(X) + \Phi_1(Z) \|_2 \cdot \| B_2\left(F_2(Z,W) - F_2(X,Y) \right) \|_2 \end{aligned}\right\} \cdot \quad (3.4.12a)$$

The corresponding inequality for the map $R_2(X,Y)$ is

$$\left.\begin{aligned} &\| R_2(X,Y) - R_2(Z,W) \|_2 \\ &\leqslant \frac{\| \Phi_2(Y) A_2(X,Y) + \Phi_2(W) A_2(Z,W) \|_2}{2} \cdot \| B_1\left(F_1(Z,W) - F_1(X,Y) \right) \|_2 + \\ &\frac{\| A_2(X,Y) + A_2(Z,W) \|_2^2}{4} \cdot \| \Phi_2(Y) - \Phi_2(W) \|_2 + \\ &\frac{\| A_2(X,Y) + A_2(Z,W) \|_2}{4} \cdot \| \Phi_2(Y) + \Phi_2(W) \|_2 \cdot \| B_1\left(F_1(Z,W) - F_1(X,Y) \right) \|_2 \end{aligned}\right\} \cdot \quad (3.4.12b)$$

In order to produce the conditions for which (3.4.6) holds in the region $\Delta \stackrel{\Delta}{=} B_{\bar{\delta}_1} \times B_{\bar{\delta}_2}$, take

$$R_i \stackrel{\Delta}{=} \frac{1}{\epsilon_i} I, \ \epsilon_i > 0, \ i \in \{1,2\} \quad (3.4.13)$$

and show that for any and all positive-semidefinite symmetric $\tilde{R}_i$ with $\underline{\sigma}(\tilde{R}_i) \geqslant \frac{1}{\bar{\epsilon}_i(\bar{\delta}_i)} > 0$,

$$\| R_i(X,Y) - R_i(Z,W) \|_2 \leqslant \alpha(\bar{\delta}_i) \cdot \left| \| X - Z \|_2 + \| Y - W \|_2 \right| \quad (3.4.14)$$

where $(X,Y),(Z,W) \in \Delta$, $0 < \alpha(\bar{\delta}_i) < 1$, and $\bar{\epsilon}_i(\bar{\delta}_i)$ is yet to be determined. Then, the contraction mapping argument will become obvious.

Therefore, some useful facts are stated. The proofs may be found in Appendix B. The following result introduces the important variables ν_1 and ν_2.

Fact **3.2 :** Given any $X, Y \in B_{\delta_i}$ and $R_i \triangleq \frac{1}{\epsilon_i} I$, $\epsilon_i > 0$, then whenever $\epsilon_i < \frac{1}{\delta_i \| B_i \|_2^2}$ holds for all $i \in \{1,2\}$, $\| \Phi_i(X) - \Phi_i(Y) \|_2 \leq \frac{1}{(1-\nu_i)^2} \| X - Y \|_2$ where $\nu_i \triangleq \epsilon_i \delta_i \| B_i \|_2^2 < 1$.

Proof **:** See Appendix B

□

Fact **3.3 :** Given any $(X, Y) \in \Delta \triangleq B_{\bar{\delta}_1} \times B_{\bar{\delta}_2}$ and $R_i \triangleq \frac{1}{\epsilon_i} I$, $\epsilon_i > 0$, then whenever $0 < \epsilon_i < \bar{\epsilon}_i (\bar{\delta}_i) \triangleq \frac{1}{\bar{\delta}_i \| B_i \|_2^2}$ holds for all $i \in \{1,2\}$, it follows that

$$\rho_1(X, Y) = \| A_1(X, Y) \|_2 \leq \xi \frac{1 + \bar{\nu}_2}{1 - \bar{\nu}_1 \bar{\nu}_2} \triangleq \xi \bar{\kappa}_1 , \quad \bar{\kappa}_1 \geq 1$$

$$\rho_2(X, Y) = \| A_2(X, Y) \|_2 \leq \xi \frac{1 + \bar{\nu}_1}{1 - \bar{\nu}_1 \bar{\nu}_2} \triangleq \xi \bar{\kappa}_2 , \quad \bar{\kappa}_2 \geq 1$$

where $\bar{\nu}_i \triangleq \epsilon_i \bar{\delta}_i \| B_i \|_2^2$ and $\xi \triangleq \| A \|_2$.

Proof **:** See Appendix B

□

Consider (3.4.12a) and (3.4.12b) where $(X, Y), (Z, W) \in \Delta$. Consequently, (3.4.12a) coupled with Facts 3.1, 3.2, and 3.3 yields :

$$\begin{aligned}
&\| R_1(X, Y) - R_1(Z, W) \|_2 \\
&\quad \leq \frac{\| X \|_2 \cdot \| A_1(X, Y) \|_2 + \| Z \|_2 \cdot \| A_1(Z, W) \|_2}{2} \cdot \| B_2 (F_2(Z, W) - F_2(X, Y)) \|_2 + \\
&\quad \frac{(2 \xi \bar{\kappa}_1)^2}{4 (1 - \bar{\nu}_1)^2} \cdot \| X - Z \|_2 + \\
&\quad \frac{2 \xi \bar{\kappa}_1}{4} \cdot \left[\| X \|_2 + \| Z \|_2 \right] \cdot \| B_2 (F_2(Z, W) - F_2(X, Y)) \|_2 \\
&\quad \leq \xi \bar{\delta}_1 \bar{\kappa}_1 \cdot \| B_2 (F_2(Z, W) - F_2(X, Y)) \|_2 + \left[\frac{\xi \bar{\kappa}_1}{1 - \bar{\nu}_1} \right]^2 \cdot \| X - Z \|_2 + \\
&\quad \xi \bar{\delta}_1 \bar{\kappa}_1 \cdot \| B_2 (F_2(Z, W) - F_2(X, Y)) \|_2
\end{aligned}$$

$$\leq 2\xi\overline{\delta}_1\overline{\kappa_1}\,\|B_2(F_2(Z,W)-F_2(X,Y))\|_2 + \left|\frac{\xi\overline{\kappa_1}}{1-\overline{\nu_1}}\right|^2\cdot\|X-Z\|_2 \Bigg\}. \quad (3.4.15)$$

and (3.4.12b) becomes

$$\|R_2(X,Y)-R_2(Z,W)\|_2$$
$$\leq 2\xi\overline{\delta}_2\overline{\kappa_2}\cdot\|B_1(F_1(Z,W)-F_1(X,Y))\|_2 + \left|\frac{\xi\overline{\kappa_2}}{1-\overline{\nu_2}}\right|^2\cdot\|Y-W\|_2 \Bigg\}. \quad (3.4.16)$$

Equations (3.4.15)-(3.4.16) are nearing the form of (3.4.14). The next result allows the contraction mapping argument to be completed. The details of the proof may be found in Appendix B.

Theorem **3.2** : Lipschitz Constants for the Feedbacks

Given $(X,Y),(Z,W)\in\Delta$, then

$$\|B_1(F_1(Z,W)-F_1(X,Y))\|_2 \leq \alpha_{11}\|X-Z\|_2 + \alpha_{12}\|Y-W\|_2$$
$$\|B_2(F_2(Z,W)-F_2(X,Y))\|_2 \leq \alpha_{21}\|X-Z\|_2 + \alpha_{22}\|Y-W\|_2$$

where $\alpha_{11}=\dfrac{\xi\overline{\nu_1}\overline{\kappa_1}\overline{\kappa_2}}{\overline{\delta}_1}$, $\alpha_{12}=\dfrac{\xi\overline{\nu_1}\overline{\nu_2}\overline{\kappa_1}\overline{\kappa_2}}{\overline{\delta}_2}$, $\alpha_{21}=\dfrac{\xi\overline{\nu_1}\overline{\nu_2}\overline{\kappa_1}\overline{\kappa_2}}{\overline{\delta}_1}$, and $\alpha_{22}=\dfrac{\xi\overline{\nu_2}\overline{\kappa_1}\overline{\kappa_2}}{\overline{\delta}_2}$.

Proof : See Appendix B

□

Making use of Theorem 3.2, (3.4.15) may be rewritten as :

$$\|R_1(X,Y)-R_1(Z,W)\|_2$$
$$\leq 2\xi\overline{\delta}_1\overline{\kappa_1}\left\{\alpha_{11}\|X-Z\|_2+\alpha_{12}\|Y-W\|_2\right\} + \left|\frac{\xi\overline{\kappa_1}}{1-\overline{\nu_1}}\right|^2\cdot\|X-Z\|_2$$
$$= \left\{2\xi\overline{\delta}_1\overline{\kappa_1}\alpha_{11} + \left|\frac{\xi\overline{\kappa_1}}{1-\overline{\nu_1}}\right|^2\right\}\cdot\|X-Z\|_2 + 2\xi\overline{\delta}_1\overline{\kappa_1}\alpha_{12}\cdot\|Y-W\|_2$$

$$= \left| 2\,\xi^2\,\overline{\nu_1}\,(\overline{\kappa_1})^2\,\overline{\kappa_2} + \frac{\xi^2\,(\overline{\kappa_1})^2}{1-\overline{\nu_1}^2} \right| \cdot \| X - Z \|_2 + 2\,\xi^2 \left| \frac{\overline{\delta}_1}{\overline{\delta}_2} \right| \overline{\nu_1}\,\overline{\nu_2}\,(\overline{\kappa_1})^2\,\overline{\kappa_2} \cdot \| Y - W \|_2$$

$$= \xi^2\,(\overline{\kappa_1})^2 \left| 2\,\overline{\nu_1}\,\overline{\kappa_2} + \frac{1}{(1-\overline{\nu_1})^2} \right| \cdot \| X - Z \|_2 \qquad (3.4.17a)$$

$$+ 2\,\xi^2 \left| \frac{\overline{\delta}_1}{\overline{\delta}_2} \right| \overline{\nu_1}\,\overline{\nu_2}\,(\overline{\kappa_1})^2\,\overline{\kappa_2} \cdot \| Y - W \|_2 \ .$$

Likewise, (3.4.16) reduces to

$$\| R_2\,(X, Y) - R_2\,(Z, W) \|_2$$

$$\leq 2\,\xi^2 \left| \frac{\overline{\delta}_2}{\overline{\delta}_1} \right| \overline{\nu_1}\,\overline{\nu_2}\,\overline{\kappa_1}\,(\overline{\kappa_2})^2 \cdot \| X - Z \|_2 \qquad (3.4.17b)$$

$$+ \xi^2\,(\overline{\kappa_2})^2 \left| 2\,\overline{\nu_2}\,\overline{\kappa_1} + \frac{1}{(1-\overline{\nu_2})^2} \right| \cdot \| Y - W \|_2 \ .$$

Finally, adding (3.4.17a) and (3.4.17b) yields the desired result.

$$\| R_1\,(X, Y) - R_1\,(Z, W) \|_2 + \| R_2\,(X, Y) - R_2\,(Z, W) \|_2$$

$$\leq \xi^2 \left| 2\,\overline{\nu_1}\,(\overline{\kappa_1})^2\,\overline{\kappa_2} + \frac{(\overline{\kappa_1})^2}{(1-\overline{\nu_1})^2} + 2 \left| \frac{\overline{\delta}_2}{\overline{\delta}_1} \right| \overline{\nu_1}\,\overline{\nu_2}\,\overline{\kappa_1}\,(\overline{\kappa_2})^2 \right| \cdot \| X - Z \|_2 +$$

$$\xi^2 \left| 2\,\overline{\nu_2}\,\overline{\kappa_1}\,(\overline{\kappa_2})^2 + \frac{(\overline{\kappa_2})^2}{(1-\overline{\nu_2})^2} + 2 \left| \frac{\overline{\delta}_1}{\overline{\delta}_2} \right| \overline{\nu_1}\,\overline{\nu_2}\,(\overline{\kappa_1})^2\,\overline{\kappa_2} \right| \cdot \| Y - W \|_2$$

$$\stackrel{\Delta}{=} \xi^2\,\alpha_1 \cdot \| X - Z \|_2 + \xi^2\,\alpha_2 \cdot \| Y - W \|_2$$

$$\leq \xi^2 \cdot max\,\{\alpha_1, \alpha_2\} \cdot \left(\| X - Z \|_2 + \| Y - W \|_2 \right) . \qquad (3.4.18)$$

Clearly, (3.4.18) and (3.4.6) are of the same form with

$$\alpha = \xi^2 \cdot max\,\{\alpha_1, \alpha_2\} \qquad (3.4.19)$$

Moreover, $\xi \stackrel{\Delta}{=} \| A \|_2$ can always be chosen sufficiently small enough so that $0 \leq \alpha < 1$. Hence, the existence of a non-trivial region where coupled discrete-time Riccati equations behave as a contraction mapping is established.

It is apparent that there are 3 parameters which govern the region where a contraction mapping

occurs. These are ϵ_1 , ϵ_2 , and ξ . Before concluding this section, a procedure is developed for determining those ranges of values for which a contraction mapping is guaranteed. Essentially, these quantities must be chosen small enough so that all the assumptions of the derivation remain valid. To begin with, ξ is required to be smaller than some $\overline{\xi}$ as yet to be determined. From (3.4.7) and Claim 3.1 :

$$\rho_i(X, Y) \stackrel{\Delta}{=} \| A_i(X, Y) \|_2 \leqslant \overline{\rho}_i < 1 \ . \tag{3.4.20}$$

But in view of Fact 3.3 and without loss of generality :

$$\overline{\rho}_i \stackrel{\Delta}{=} \xi \overline{\kappa}_i < 1 \tag{3.4.21}$$

provided $\xi < \min\left\{\frac{1}{\overline{\kappa}_1}, \frac{1}{\overline{\kappa}_2}\right\}$ and $0 < \epsilon_i < \overline{\epsilon}_i(\overline{\delta}_i) \stackrel{\Delta}{=} \frac{1}{\overline{\delta}_i \| B_i \|_2^2}$, i = 1, 2. However, there is a more restrictive condition imposed on ξ by (3.4.19). That is

$$\xi < \min\left\{\frac{1}{\sqrt{\alpha_1}}, \frac{1}{\sqrt{\alpha_2}}\right\} . \tag{3.4.22}$$

In fact, it is easily shown that $\min\left\{\frac{1}{\sqrt{\alpha_1}}, \frac{1}{\sqrt{\alpha_2}}\right\} \leqslant \min\left\{\frac{1}{\overline{\kappa}_1}, \frac{1}{\overline{\kappa}_2}\right\}$ with equality if and only if $\epsilon_1 = \epsilon_2 = 0$. Therefore, define $\overline{\xi} \stackrel{\Delta}{=} \min\left\{\frac{1}{\sqrt{\alpha_1}}, \frac{1}{\sqrt{\alpha_2}}\right\}$ and pick ξ such that $\xi < \overline{\xi}$.

Now, a moment's reflection reveals that, in general, the 3 parameters ϵ_1 , ϵ_2 , and ξ *cannot* be *explicitly* solved for. To see this, consider ξ which must be chosen smaller than $\overline{\xi}$. Well, $\overline{\xi}$ is a function of α_1 and α_2 which, in turn, are functions of $\overline{\delta}_1$ and $\overline{\delta}_2$ through (3.4.18) . But, $\overline{\delta}_i$ is a function of $\overline{\rho}_i$ by (3.4.8) and (3.4.21) indicates that $\overline{\rho}_i$ is a function of ξ . Hence ξ is a nonlinear function of itself. Similar arguments can be stated for ϵ_1 and ϵ_2 .

Therefore, the task at hand is to enumerate the cases where ϵ_1 , ϵ_2 , and ξ yield a contraction mapping. The following algorithm accomplishes this task.

Contraction Mapping Surface Generator Algorithm

Step 1 : For $\overline{\nu}_1$, $\overline{\nu}_2 \in (0, 1)$ but fixed do the following :

Step 2 : Compute $\overline{\kappa_1}$ and $\overline{\kappa_2}$ as defined in Fact 3.3.

Step 3 : Choose $\overline{\xi} = \min\left\{\frac{1}{\overline{\kappa_1}}, \frac{1}{\overline{\kappa_2}}\right\}$.

Step 4 : For $\xi \in (0, \overline{\xi})$, compute $\overline{\delta}_1(\xi)$ and $\overline{\delta}_2(\xi)$.

Step 5 : Compute α_1 and α_2 as defined in (3.4.18).

Step 6 : If $\xi < \min\left\{\frac{1}{\sqrt{\alpha_1}}, \frac{1}{\sqrt{\alpha_2}}\right\}$ then pick $\overline{\epsilon_1}$ and $\overline{\epsilon_2}$ to give $\overline{\nu_1}$ and $\overline{\nu_2}$ as fixed in Step 1.

Save the triple $(\overline{\epsilon_1}, \overline{\epsilon_2}, \xi)$ as a valid combination.

Step 7 : If not done then go to Step 1 else stop.

If this algorithm is implemented on a computer, then a list of possible maximal values can be compiled. Actually, the roles of $\overline{\nu_1}$ and $\overline{\nu_2}$ are completely interchangeable insofar as the norms of Q_1 and Q_2 are equal. Table 3.1 summarizes various maximal ranges of the 3 parameters.

For example, let $\| B_1 \|_2 = \| B_2 \|_2 = \| Q_1 \|_2 = \| Q_2 \|_2 = 1$. Given any $\tilde{R}_1$ and $\tilde{R}_2$ with $\underline{\sigma}(\tilde{R}_1), \underline{\sigma}(\tilde{R}_2) > \frac{1}{0.4800} = 2.0833$ and any A with $\xi = \overline{\sigma}(A) < 0.1$, then the corresponding coupled discrete-time Riccati equation iterations will be a contraction mapping for all initial conditions (X_0, Y_0) where $\overline{\sigma}(X_0), \overline{\sigma}(Y_0) \leqslant 1.04$. Note that if dynamic programming were used to solve these equations, then the initial condition would be $(X_0, Y_0) = (Q_1, Q_2)$ and hence the requirement that the initial conditions lie in a ball of radius 1.04 times the magnitude of Q_i is automatically satisfied. Furthermore, the contraction mapping constant defined by (3.4.19) is $\tilde{\alpha} < (0.1)^2 \cdot 28.0 = 0.28$. Obviously, the requirement that $\overline{\sigma}(A) < 0.1$ is a conservative one. This is due, in part, to the large discretization increment of Table 3.1.

3.4.3 A Numerical Example

In this subsection, we construct a numerical example that illustrates the contraction mapping derived in the last subsection. Consider the system (2.1.1)-(2.1.2) with performance indices (2.1.3). Let $C_1 = C_2 = E = S_1 = S_2 = \mathbf{I}$. Then choose

Table 3.1. Normalized* Maximal Ranges of Values for Which the Coupled Riccatis Are Contractions

ν_1	ν_2	ξ	δ_1	δ_2	ϵ_1	ϵ_2	α_1	α_2
0.10	0.10	0.70	2.53e+00	2.53e+00	0.0395	0.0395	1.83e+00	1.83e+00
0.10	0.20	0.60	2.17e+00	1.83e+00	0.0460	0.1093	2.24e+00	2.67e+00
0.10	0.30	0.50	1.82e+00	1.47e+00	0.0551	0.2035	2.71e+00	3.81e+00
0.10	0.40	0.40	1.52e+00	1.27e+00	0.0660	0.3160	3.24e+00	5.41e+00
0.10	0.50	0.30	1.29e+00	1.14e+00	0.0776	0.4397	3.84e+00	7.81e+00
0.10	0.60	0.20	1.13e+00	1.06e+00	0.0884	0.5671	4.52e+00	1.18e+01
0.10	0.70	0.20	1.15e+00	1.06e+00	0.0866	0.6608	5.24e+00	1.97e+01
0.10	0.80	0.10	1.04e+00	1.01e+00	0.0962	0.7886	6.08e+00	4.10e+01
0.20	0.20	0.50	1.64e+00	1.64e+00	0.1219	0.1219	3.38e+00	3.38e+00
0.20	0.30	0.40	1.44e+00	1.35e+00	0.1388	0.2218	4.22e+00	4.99e+00
0.20	0.40	0.30	1.26e+00	1.18e+00	0.1583	0.3388	5.21e+00	7.31e+00
0.20	0.50	0.30	1.33e+00	1.19e+00	0.1500	0.4200	6.35e+00	1.09e+01
0.20	0.60	0.20	1.15e+00	1.08e+00	0.1736	0.5554	7.73e+00	1.68e+01
0.20	0.70	0.10	1.04e+00	1.02e+00	0.1922	0.6864	9.34e+00	2.86e+01
0.20	0.80	0.10	1.05e+00	1.02e+00	0.1908	0.7837	1.12e+01	6.02e+01
0.30	0.30	0.30	1.23e+00	1.23e+00	0.2449	0.2449	6.44e+00	6.44e+00
0.30	0.40	0.30	1.29e+00	1.24e+00	0.2317	0.3214	8.21e+00	9.77e+00
0.30	0.50	0.20	1.14e+00	1.10e+00	0.2626	0.4532	1.04e+01	1.50e+01
0.30	0.60	0.20	1.18e+00	1.11e+00	0.2543	0.5397	1.31e+01	2.39e+01
0.30	0.70	0.10	1.05e+00	1.03e+00	0.2861	0.6810	1.64e+01	4.15e+01
0.30	0.80	0.10	1.06e+00	1.03e+00	0.2832	0.7766	2.04e+01	8.90e+01
0.40	0.40	0.20	1.13e+00	1.13e+00	0.3556	0.3556	1.29e+01	1.29e+01
0.40	0.50	0.20	1.16e+00	1.14e+00	0.3438	0.4388	1.69e+01	2.05e+01
0.40	0.60	0.10	1.05e+00	1.04e+00	0.3823	0.5796	2.22e+01	3.37e+01
0.40	0.70	0.10	1.06e+00	1.04e+00	0.3777	0.6735	2.91e+01	6.07e+01
0.50	0.50	0.10	1.04e+00	1.04e+00	0.4800	0.4800	2.80e+01	2.80e+01
0.50	0.60	0.10	1.06e+00	1.05e+00	0.4739	0.5724	3.83e+01	4.81e+01
0.50	0.70	0.10	1.07e+00	1.06e+00	0.4658	0.6627	5.27e+01	8.99e+01
0.60	0.60	0.10	1.07e+00	1.07e+00	0.5625	0.5625	6.91e+01	6.91e+01

* Assumes $\| Q_1 \|_2 = \| Q_2 \|_2 = 1 = \| B_1 \|_2 = \| B_2 \|_2$

$$A = \begin{vmatrix} 0.4755 & 0.0459 & -1.13e-4 \\ 0.0459 & 0.345 & -7.175e-5 \\ -1.13e-4 & -7.175e-5 & 0.25 \end{vmatrix},$$

$$B_1 = \begin{vmatrix} 9.87e2 \\ 1.23 \\ -1.01e-3 \end{vmatrix} \quad \text{and} \quad B_2 = \begin{vmatrix} 1.37 \\ -1.0e-3 \\ 1.0e-5 \end{vmatrix}.$$

The eigenvalues of A are 0.49, 0.33, and 0.25. Next, let $R_1 = R_2 = 11.0$. From Table 3.1 it is observed that this case falls within the region of contraction mapping. To illustrate the contraction mapping behavior, this problem is run for 14 iterations using the L-A-S operators described in

Chapter 4. A full listing of this run may be found in Appendix C. The important results are summarized in Table 3.2. The values of K_1 and K_2 at the end of the iterations are

$$K_1 = \begin{bmatrix} 1.00233 & 1.76897e-2 & -3.43357e-5 \\ 1.76897e-2 & 1.13453 & -5.14615e-5 \\ -3.43357e-5 & -5.14615e-5 & 1.06667 \end{bmatrix}$$

$$K_2 = \begin{bmatrix} 1.00233 & 1.76895e-2 & -3.43351e-5 \\ 1.76895e-2 & 1.13453 & -5.14618e-5 \\ -3.43351e-5 & -5.14618e-5 & 1.06667 \end{bmatrix}.$$

Observe that the iterations converge rapidly. After 14 iterations, K_1(k) and K_2(k) are changing by no more than 1.0e-13. Even more remarkable is the fact that α is almost constant throughout the iterations. One final thing to note is that the predicted contraction mapping constant is $(0.5)^2 \cdot 3.38 = 0.845$, while the observed constant is almost an order of magnitude less than that. In general, this large difference occurs because the inequalities used in proving the contraction mapping are not all tight simultaneously.

Since the contraction mapping argument guarantees convergence to a unique fixed point and the values of K_1 and K_2 are accurate to 1.0e-13, then we conclude that these values for K_1 and K_2

Table 3.2. Results of Contraction Mapping Iterations

Iteration #	ΔK_1	ΔK_2	α
0	1.00000e+00	1.00000e+00	1.20666e-01
1	1.20666e-01	1.20666e-01	1.18329e-01
2	1.42782e-02	1.42782e-02	1.18053e-01
3	1.68558e-03	1.68558e-03	1.18020e-01
4	1.98932e-04	1.98933e-04	1.18016e-01
5	2.34772e-05	2.34773e-05	1.18016e-01
6	2.77068e-06	2.77069e-06	1.18016e-01
7	3.26983e-07	3.26986e-07	1.18016e-01
8	3.85891e-08	3.85894e-08	1.18016e-01
9	4.55412e-09	4.55416e-09	1.18016e-01
10	5.37457e-10	5.37463e-10	1.18016e-01
11	6.34284e-11	6.34291e-11	1.18016e-01
12	7.48554e-12	7.48565e-12	1.18014e-01
13	8.83404e-13	8.83404e-13	1.18020e-01
14	1.04246e-13	1.04273e-13	1.18080e-01

satisfy the Algebraic Riccati Equations (ARE) to 13 significant digits. Thus, for some cases, the coupled Riccati iteration algorithm provides a method for obtaining the solution to two coupled discrete-time AREs. This solution may also be the solution to the infinite-horizon LQ Nash game problem.

CHAPTER 4

L-A-S OPERATORS FOR SINGLE AND COUPLED DISCRETE-TIME RICCATI ITERATIONS

This chapter describes the L-A-S [32-36] operators created for the task of iterating single and coupled discrete-time Riccati equations. There are a total of six new operators to the L-A-S package. As presented in this chapter, they are SYST, LQ, DRE, GAME, LQNG, and MLTR. Single Riccati iterations are addressed first. Then the coupled (game) case is described. Operators SYST and DRE fall into the first category, while operators GAME, LQNG, and MLTR fall into the second category. L-A-S operator LQ is used for both single and coupled Riccati iterations.

First, a brief description of the operator is given. Next, the corresponding excerpt from the L-A-S Help-File is presented. Then a typical example of the usage of the operator is demonstrated.

4.1 L–A–S Operator SYST

The L-A-S operator named SYST is used to define a linear shift-invariant descriptor system. Often, it is the first operator issued to the L-A-S interpreter when a linear quadratic regulator problem is being studied.

L–A–S Help–File Description

SYST - Descriptor SYSTem description
Syntax : A, B, C [, D [, E]] (SYST) =
Input Data : A [N,N] , B [N,P] , C [M,N] ,
D [M,P] , E [N,N] (these last two arrays optional)
Options : E, L, T
Description : Identifies the following discrete-time descriptor system

$$E\,x(k+1) = A\,x(k) + B\,u(k)$$

$$y(k) = C\,x(k) + D\,u(k)$$

Note : If E is omitted, it is assumed to be the identity matrix.
If D is omitted, it is assumed to be a zero matrix.

Example of Usage

An example of the usage of operator SYST is given below. The following segment is an L-A-S program that identifies a discrete-time descriptor system.

```
: Descriptor System Identification
(inp)=a,b,c,d,e
a,b,c,d,e(out)=
a,b,c,d,e(syst)=
```

Notice that the program is completely generic in the sense that the dimensions of the matrices a, b, c, d, and e are not specified. This is an important feature of the L-A-S language. The same program can be run over and over again using different matrices (of different order) each time. If this program is executed for a simple second-order system, the following output would result.

```
>: Descriptor System Identification

>(inp)=a,b,c,d,e

*** Matrix a ***
Enter the dimensions of this matrix.  >2,2

Matrix : a     Enter  C,D,E,I,N,P,R,Z or H for Help.  >R
ROW 1 >1,2
ROW 2 >3,4

*** Matrix b ***
Enter the dimensions of this matrix.  >2,1

Matrix : b     Enter  C,D,E,I,N,P,R,Z or H for Help.  >C
COL 1 >1,0

*** Matrix c ***
Enter the dimensions of this matrix.  >1,2

Matrix : c     Enter  C,D,E,I,N,P,R,Z or H for Help.  >R
ROW 1 >0,1

*** Matrix d ***
Enter the dimensions of this matrix.  >1,1

Enter the scalar : d     >0

*** Matrix e ***
Enter the dimensions of this matrix.  >2,2

Matrix : e     Enter  C,D,E,I,N,P,R,Z or H for Help.  >I
```

```
>a,b,c,d,e(out)=

     a
1.000   2.000
3.000   4.000

     b
1.000
0.

     c
0.     1.000

     d
0.

     e
1.000   0.
0.      1.000

>a,b,c,d,e(syst)=
```

The matrices d and e are optional as indicated in the Help-File description. However, if matrix e is specified then matrix d must also be provided to serve as a placeholder. That is, a statement such as

a,b,c,,e(syst)=

is not permitted.

4.2 L–A–S Operator LQ

The L-A-S operator named LQ is used to define the weighting matrices in a discrete-time system or game problem. Often, it is the second operator issued to the L-A-S interpreter when a linear quadratic regulator or Nash game problem is being studied.

L–A–S Help–File Description

```
LQ  - Linear Quadratic weighting matrices for system or game theoretic problems
  Syntax       : Q1, R1 [, Q2, R2] (LQ) =
  Input Data   : Q1 [N,N] , R1 [P1,P1] , Q2 [N,N] , R2 [P2,P2]
                 (These last two arrays only required for game)
  Options      : E, L, T
  Description  : Identifies the weighting matrices in a linear-quadratic discrete-time
                 system or game problem
```

Example of Usage

An example of the usage of operator LQ is given below. Consider the discrete-time system defined in Section 4.1. The following segment is an L-A-S program that identifies the same system and then defines two weighting matrices in preparation for the study of a linear quadratic regulator problem. Rather than inputting the matrices from the keyboard, the RDF (Read Data File) operator is used.

```
: Linear Quadratic Regulator Problem
: System and Weighting Matrix Identification
;
: System Definition
(rdf)=a,b,c
a,b,c(out)=
a,b,c(syst)=
;
: Weighting Matrices Definition
(rdf)=q,r
q,r(out)=
q,r(lq)=
```

If this program is run using the second-order system of Section 4.1, the following output results.

```
>: Linear Quadratic Regulator Problem

>: System and Weighting Matrix Identification

>;

>: System Definition

>(rdf)=a,b,c

Enter name of the Data File (DF) for matrix a    >syst
Opening file named : syst.DF
Reading array named : a
Reading array named : b
Reading array named : c

>a,b,c(out)=

      a
1.000   2.000
3.000   4.000

      b
1.000
0.
```

```
      c
0.     1.000

>a,b,c(syst)=

>;

>; Weighting Matrices Definition

>(rdf)=q,r

Enter name of the Data File (DF) for matrix q    >lq
Opening file named : lq.DF
Reading array named : q
Reading array named : r

>q,r(out)=

      q
1.000   0.
0.      2.000

      r
1.000

>q,r(lq)=
```

4.3 L–A–S Operator DRE

The L-A-S operator named DRE is used to iterate a single discrete-time Riccati equation.

L–A–S Help–File Description

DRE - Discrete-time Riccati Equation iteration
Syntax : K (DRE) = KNEW
Input Data : K [N,N]
Output Data : KNEW [N,N]
Options : E, L, T
Description : Assuming that operators SYST and LQ have been issued, this operator performs one iteration of the discrete-time Riccati equation defined by

$$E^T \, \mathrm{KNEW} \, E = A^T (\mathrm{K} - \Psi^T \Gamma \Psi) A + Q$$

where $\Gamma = R + B^T K B$, and $\Psi = (\Gamma)^{-1} B^T K$.
A, B, and E are identified by the SYST operator. Q and R are identified by the LQ operator.

Example of Usage

An example of the usage of operator DRE is given below. Consider the discrete-time system defined in Section 4.1. The following segment is an L-A-S program that identifies the same system and then defines two weighting matrices in preparation for the study of a linear quadratic regulator problem. Rather than inputting the matrices from the keyboard, the RDF (Read Data File) operator is used. Since the matrix E is equal to the identity for this problem, the terminal constraint (2.1.14) reduces to $K = Q$ which is performed in step 13 below.

```
 1 :Linear Quadratic Regulator Problem
 2 ;
 3 :System and Weighting Matrix Identification
 4 ; --- System Definition ---
 5 (rdf)=a,b,c
 6 a,b,c(out)=
 7 a,b,c(syst)=
 8 ; --- Weighting Matrices Definition ---
 9 (rdf)=q,r
10 q,r(out)=
11 q,r(lq)=
12 :Initialization
13 q(mcp)=k
14 1(dsc)=one
15 ;
16 "Enter the total number of iterations to perform. (A scalar)"
17 (inp)=num
18 ;
19 ; Main Loop
20 Z:k(out)=
21 k(dre)=knew
22 knew(mcp)=k
23 num,one(-)=num
24 num(if)=Z
25 ;
26 ; Done. Print out final k.
27 k(out)=
28 (stop)=
```

If this program is run for five iterations using the second-order system of Section 4.1, the following output results.

```
> :Linear Quadratic Regulator Problem

> :

> :System and Weighting Matrix Identification
```

```
>; --- System Definition ---

>(rdf)=a,b,c

Enter name of the Data File (DF) for matrix a    >syst
Opening file named : syst.DF
Reading array named : a
Reading array named : b
Reading array named : c

>a,b,c(out)=

      a
1.000   2.000
3.000   4.000

      b
1.000
0.

      c
0.      1.000

>a,b,c(syst)=

>; --- Weighting Matrices Definition ---

>(rdf)=q,r

Enter name of the Data File (DF) for matrix q    >lq
Opening file named : lq.DF
Reading array named : q
Reading array named : r

>q,r(out)=

      q
1.000   0.
0.      2.000

      r
1.000

>q,r(lq)=

>;Initialization

>q(mcp)=k

>1(dsc)=one

>;
```

```
>"Enter the total number of iterations to perform. (A scalar)"

Enter the total number of iterations to perform. (A scalar)

>(inp)=num

*** Matrix num ***
Enter the dimensions of this matrix.  >1,1

Enter the scalar : num   >5

>:

>: Main Loop

>Z:k(out)=

      k
1.000   0.
0.      2.000

>k(dre)=knew

>knew(mcp)=k

>num.one(-)=num

>num(if)=Z

>Z:k(out)=

      k
19.500  25.000
25.000  36.000

>k(dre)=knew

>knew(mcp)=k

>num.one(-)=num

>num(if)=Z

>Z:k(out)=

      k
58.878  80.244
80.244 113.512

>k(dre)=knew

>knew(mcp)=k
```

```
>num,one(-)=num

>num(if)=Z

>Z:k(out)=

      k
63.804  87.075
87.075 122.984

>k(dre)=knew

>knew(mcp)=k

>num,one(-)=num

>num(if)=Z

>Z:k(out)=

      k
63.918  87.234
87.234 123.208

>k(dre)=knew

>knew(mcp)=k

>num,one(-)=num

>num(if)=Z

>:

>: Done. Print out final k.

>k(out)=

      k
63.922  87.240
87.240 123.216

>(stop)=
```

Notice the speed with which the Riccati iterations converge. The value of k is settled to three significant digits in only five iterations.

4.4 L–A–S Operator GAME

The L-A-S operator named GAME is used to define a linear shift-invariant descriptor game. Often, it is the first operator issued to the L-A-S interpreter when a linear quadratic descriptor Nash game problem is being studied.

L–A–S Help–File Description

GAME - 2-player GAME description
Syntax : A, B1, B2, C1, C2 [,E] (GAME) =
Input Data : A [N,N] , B1 [N,P1] , B2 [N,P2] ,
C1 [M1,N] , C2 [M2,N] , E [N,N]
Matrix E is optional.
Options : E, L, T
Description : Identifies the following discrete-time game :

$$E\,x(k+1) = A\,x(k) + B_1 u_1(k) + B_2 u_2(k)$$

$$y_1(k) = C_1 x(k)$$

$$y_2(k) = C_2 x(k)$$

Note : If E is omitted, it is assumed to be the identity matrix.

Example of Usage

An example of the usage of operator GAME is given below. The following L-A-S program segment identifies a discrete-time descriptor game.

```
; Descriptor Game Identification
(rdf)=a,b1,b2,c1,c2
a,b1,b2,c1,c2(out)=
a,b1,b2,c1,c2(game)=
```

If this program is executed for a third-order game problem, the following output would result.

```
>: Descriptor Game Identification

>(rdf)=a,b1,b2,c1,c2

Enter name of the Data File (DF) for matrix a    >ab
Opening file named : ab.DF
Reading array named : a
Reading array named : b1
Reading array named : b2
```

```
Enter name of the Data File (DF) for matrix c1   >cc
Opening file named : cc.DF
Reading array named : c1
Reading array named : c2

 >a,b1,b2,c1,c2(out)=

      a
 0.435  -1.401  -0.896
-0.172  -0.569   1.391
-1.655   0.008   0.134

      b1
 1.000   2.000
 3.000   4.000
 5.000   6.000

      b2
 1.000
 1.000
 0.

      c1
 1.000   0.      0.
 0.      1.000   0.
 0.      0.      1.000

      c2
 1.000   0.      0.
 0.      1.000   0.
 0.      0.      1.000

 >a,b1,b2,c1,c2(game)=
```

4.5 L–A–S Operator LQNG

The L-A-S operator named LQNG is used to iterate two coupled discrete-time Riccati equations.

L–A–S Help–File Description

```
LQNG - Linear Quadratic Nash Game (Coupled Riccati Iterations)
  Syntax       : K1, K2 (LQNG) = K1N, K2N [, F1 [, F2] ]
  Input Data   : K1 [N,N] , K2 [N,N]
  Output Data  : K1N [N,N] , K2N [N,N] , F1 [P1,N] , F2 [P2,N]
                 Arrays F1 and F2 are optional.
  Options      : E, L, T
  Description  : Assuming that operators GAME and LQ have been issued,
                 this operator performs one iteration of two coupled
                 discrete-time Riccati equations.  K1N and K2N are the
```

new Riccati gain matrices. F1 and F2 (if provided)
are the corresponding feedback matrices.

Example of Usage

An example of the usage of operator LQNG is given below. Consider the discrete-time game defined in Section 4.4. The following segment is an L-A-S program that identifies the same system and then defines two weighting matrices in preparation for the study of a linear quadratic Nash game. Rather than inputting the matrices from the keyboard, the RDF (Read Data File) operator is used. Since the matrix E is equal to the identity for this problem, the terminal constraint (2.1.14) reduces to $K_i = Q_i$ which is performed in step 10 below.

```
 1 ; Linear Quadratic Nash Game
 2 (rdf)=a,b1,b2,c1,c2
 3 a,b1,b2,c1,c2(out)=
 4 a,b1,b2,c1,c2(game)=
 5 (rdf)=r1,r2,s1,s2
 6 r1,r2,s1,s2(out)=
 7 c1(t),s1(*),c1(*)=q1
 8 c2(t),s2(*),c2(*)=q2
 9 q1,r1,q2,r2(lq)=
10 q1,q2(mcp)=k1,k2
11 ;
12 1(dsc)=one
13 "Enter the total number of stages in this game."
14 (inp)=ii
15 ; Main Loop
16 a:k1,k2(out)=
17 k1,k2(lqng)=k1n,k2n
18 k1n,k2n(out)=
19 k1n,k2n(mcp)=k1,k2
20 ii,one(-)=ii
21 ii(if)=a
```

If this program is run for five iterations using the third-order system of Section 4.4, the following output results.

```
>; Linear Quadratic Nash Game

>(rdf)=a,b1,b2,c1,c2

Enter name of the Data File (DF) for matrix a    >ab
Opening file named : ab.DF
Reading array named : a
Reading array named : b1
```

```
Reading array named : b2

Enter name of the Data File (DF) for matrix c1   >cc
Opening file named : cc.DF
Reading array named : c1
Reading array named : c2

 >a,b1,b2,c1,c2(out)=

      a
 0.435  -1.401  -0.896
-0.172  -0.569   1.391
-1.655   0.008   0.134

      b1
 1.000   2.000
 3.000   4.000
 5.000   6.000

      b2
 1.000
 1.000
 0.

      c1
 1.000   0.      0.
 0.      1.000   0.
 0.      0.      1.000

      c2
 1.000   0.      0.
 0.      1.000   0.
 0.      0.      1.000

 >a,b1,b2,c1,c2(game)=

 >(rdf)=r1,r2,s1,s2

Enter name of the Data File (DF) for matrix r1   >rs
Opening file named : rs.DF
Reading array named : r1
Reading array named : r2
Reading array named : s1
Reading array named : s2

 >r1,r2,s1,s2(out)=

      r1
 1.000   0.
 0.      1.000

      r2
 1.000
```

```
     s1
1.000  0.     0.
0.     1.000  0.
0.     0.     1.000

     s2
1.000  0.     0.
0.     1.000  0.
0.     0.     1.000

>c1(t),s1(*),c1(*)=q1
 #1,s1(*),c1(*)=q1
 #2,c1(*)=q1

>c2(t),s2(*),c2(*)=q2
 #1,s2(*),c2(*)=q2
 #2,c2(*)=q2

>q1,r1,q2,r2(lq)=

>q1,q2(mcp)=k1,k2

>;

>1(dsc)=one

>"Enter the total number of stages in this game."

Enter the total number of stages in this game.

 >(inp)=ii

*** Matrix ii ***
Enter the dimensions of this matrix.  >1,1

Enter the scalar : ii    >5

 >; Main Loop

 >a:k1,k2(out)=

      k1
 1.000  0.     0.
 0.     1.000  0.
 0.     0.     1.000

      k2
 1.000  0.     0.
 0.     1.000  0.
 0.     0.     1.000

 >k1,k2(lqng)=k1n,k2n
```

```
>k1n,k2n(out)=

      k1n
 1.236  -0.229   0.003
-0.229   1.569   0.817
 0.003   0.817   3.291

      k2n
 1.393  -0.432   0.090
-0.432   1.771   0.709
 0.090   0.709   3.235

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>a:k1,k2(out)=

      k1
 1.236  -0.229   0.003
-0.229   1.569   0.817
 0.003   0.817   3.291

      k2
 1.393  -0.432   0.090
-0.432   1.771   0.709
 0.090   0.709   3.235

>k1,k2(lqng)=k1n,k2n

>k1n,k2n(out)=

      k1n
 1.242  -0.238  -0.009
-0.238   1.715   1.192
-0.009   1.192   4.282

      k2n
 1.402  -0.418   0.159
-0.418   1.971   1.302
 0.159   1.302   5.081

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>a:k1,k2(out)=

      k1
```

```
 1.242  -0.238  -0.009
-0.238   1.715   1.192
-0.009   1.192   4.282

      k2
 1.402  -0.418   0.159
-0.418   1.971   1.302
 0.159   1.302   5.081

>k1,k2(lqng)=k1n,k2n

>k1n,k2n(out)=

      k1n
 1.247  -0.238   0.007
-0.238   1.710   1.181
 0.007   1.181   4.304

      k2n
 1.403  -0.416   0.168
-0.416   1.977   1.321
 0.168   1.321   5.155

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>a:k1,k2(out)=

      k1
 1.247  -0.238   0.007
-0.238   1.710   1.181
 0.007   1.181   4.304

      k2
 1.403  -0.416   0.168
-0.416   1.977   1.321
 0.168   1.321   5.155

>k1,k2(lqng)=k1n,k2n

>k1n,k2n(out)=

      k1n
 1.247  -0.238   0.006
-0.238   1.714   1.192
 0.006   1.192   4.330

      k2n
 1.403  -0.416   0.169
-0.416   1.976   1.320
```

```
0.169   1.320   5.155

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>a:k1,k2(out)=

      k1
 1.247  -0.238   0.006
-0.238   1.714   1.192
 0.006   1.192   4.330

      k2
 1.403  -0.416   0.169
-0.416   1.976   1.320
 0.169   1.320   5.155

>k1,k2(lqng)=k1n,k2n

>k1n,k2n(out)=

      k1n
 1.247  -0.238   0.006
-0.238   1.714   1.192
 0.006   1.192   4.330

      k2n
 1.403  -0.416   0.169
-0.416   1.976   1.320
 0.169   1.320   5.156

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a
```

4.6 L−A−S Operator MLTR

The L-A-S operator named MLTR is used to iterate two coupled discrete-time Riccati equations where multirates are involved.

L−A−S Help−File Description

```
MLTR - MuLTiRate nash game (Multirate Coupled Riccati Iterations)
  Syntax       : K1, K2, K, N (MLTR) = K1N, K2N [, F1 [, F2] ]
  Input Data   : K1 [N,N] , K2 [N,N] , K [1,1] , N [1,1]
```

Output Data : K1N [N,N] . K2N [N,N] . F1 [P1,N] . F2 [P2,N]
Arrays F1 and F2 are optional.
Options : E, L, T
Description : Assuming that operators GAME and LQ have been issued, this operator performs one iteration of two coupled discrete-time Riccati equations where multirates are involved. K is the current time instant and N is the multirate parameter. K1N and K2N are the new Riccati gain matrices. F1 and F2 (if provided) are the corresponding feedback matrices.

Example of Usage

An example of the usage of operator MLTR is given below. Consider a third-order discrete-time system that is being controlled by two computers operating at different speeds. The computers are decentralized in that they are located at different physical places. They control the plant through a telephone hookup. DM1 is equipped with a 1200 baud modem, but DM2 has only a 300 baud modem. Thus, $N = 4$ for this problem. Because of the interface between the plant and the controllers, the input of DM2 is the all-digital control policy (2.2.3). The following L-A-S program is used to study this multirate game.

```
 1 ; Linear Quadratic Multirate Nash Game
 2 (rdf)=a,b1,b2,c1,c2
 3 a,b1,b2,c1,c2(out)=
 4 a,b1,b2,c1,c2(game)=
 5 (rdf)=s1,r1,s2,r2
 6 c1(t),s1(*),c1(*)=q1
 7 c2(t),s2(*),c2(*)=q2
 8 q1,r1,q2,r2(out)=
 9 q1,r1,q2,r2(lq)=
10 ;
11 q1,q2(mcp)=k1,k2
12 1(dsc)=one
13 "Enter the total number of stages in this game"
14 (inp)=ii
15 "Enter the multirate parameter, N"
16 (inp)=N
17 ; Main Loop
18 a:k1,k2(out)=
19 k1,k2,ii,N(mltr)=k1n,k2n,f1,f2
20 f1,f2(out,e)=
21 k1n,k2n(mcp)=k1,k2
22 ii,one(-)=ii
23 ii(if)=a
24 ;
25 k1,k2,ii,N(mltr)=k1n,k2n,f1,f2
```

```
26 f1,f2(out,e)=
```

If this program is run for six iterations using an arbitrary third-order system, the following output results.

```
>; Linear Quadratic Multirate Nash Game

>(rdf)=a,b1,b2,c1,c2

Enter name of the Data File (DF) for matrix a    >ACC1
Opening file named : ACC1.DF
Reading array named : a
Reading array named : b1
Reading array named : b2
Reading array named : c1
Reading array named : c2

>a,b1,b2,c1,c2(out)=

      a
 0.435  -1.401  -0.896
-0.172  -0.569   1.391
-1.655   0.008   0.134

      b1
 1.000
 0.
 1.000

      b2
 1.000
 1.000
 0.

      c1
 1.000  0.     0.
 0.     1.000  0.
 0.     0.     1.000

      c2
 1.000  0.     0.
 0.     1.000  0.
 0.     0.     1.000

>a,b1,b2,c1,c2(game)=

>(rdf)=s1,r1,s2,r2

Enter name of the Data File (DF) for matrix s1   >ACC2
Opening file named : ACC2.DF
Reading array named : s1
```

```
Reading array named : r1
Reading array named : s2
Reading array named : r2

>c1(t),s1(*),c1(*)=q1
 #1,s1(*),c1(*)=q1
 #2,c1(*)=q1

>c2(t),s2(*),c2(*)=q2
 #1,s2(*),c2(*)=q2
 #2,c2(*)=q2

>q1,r1,q2,r2(out)=

      q1
1.000  0.     0.
0.     1.000  0.
0.     0.     1.000

      r1
1.000

      q2
1.000  0.     0.
0.     1.000  0.
0.     0.     1.000

      r2
1.000

>q1,r1,q2,r2(lq)=

>:

>q1,q2(mcp)=k1,k2

>1(dsc)=one

>"Enter the total number of stages in this game"

Enter the total number of stages in this game

>(inp)=ii

*** Matrix ii ***
Enter the dimensions of this matrix.  >1,1

Enter the scalar : ii    >6

>"Enter the multirate parameter, N"

Enter the multirate parameter, N
```

```
>(inp)=N

*** Matrix N ***
Enter the dimensions of this matrix.  >1,1

Enter the scalar : N     >4

>; Main Loop

>a:k1,k2(out)=

      k1
1.000  0.     0.
0.     1.000  0.
0.     0.     1.000

      k2
1.000  0.     0.
0.     1.000  0.
0.     0.     1.000

>k1,k2,ii,N(mltr)=k1n,k2n,f1,f2

>f1,f2(out,e)=

      f1
-4.06588e-01   -4.64582e-01   -2.54176e-01

      f2
0.00000e+00    0.00000e+00    0.00000e+00

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>a:k1,k2(out)=

      k1
 3.461  -1.091  -1.160
-1.091   2.640   0.112
-1.160   0.112   3.562

      k2
 3.295  -1.280  -1.264
-1.280   2.424  -0.006
-1.264  -0.006   3.497

>k1,k2,ii,N(mltr)=k1n,k2n,f1,f2

>f1,f2(out,e)=
```

```
      f1
-4.91982e-01   -4.64462e-01   -5.44093e-01

      f2
 0.00000e+00    0.00000e+00    0.00000e+00

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>a:k1,k2(out)=

      k1
12.002  -5.782  -7.172
-5.782   5.706   2.611
-7.172   2.611  10.304

      k2
11.760  -6.010  -7.441
-6.010   5.214   2.712
-7.441   2.712   9.552

>k1,k2,ii,N(mltr)=k1n,k2n,f1,f2

>f1,f2(out,e)=

      f1
-6.28113e-01   -3.24123e-01   -7.37706e-01

      f2
 1.86462e+00   -1.22783e+00   -1.03105e+00

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>a:k1,k2(out)=

      k1
23.862  -8.100 -23.483
-8.100   4.268   9.051
-23.483  9.051  27.654

      k2
23.198  -9.483 -21.761
-9.483   5.342   8.857
-21.761  8.857  23.593

>k1,k2,ii,N(mltr)=k1n,k2n,f1,f2
```

```
>f1,f2(out,e)=

      f1
-1.24307e+00  -1.87479e-01   2.77446e-01

      f2
 0.00000e+00   0.00000e+00   0.00000e+00

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>a:k1,k2(out)=

      k1
112.932 -61.747 -77.912
-61.747  36.568  42.550
-77.912  42.550  57.697

      k2
107.254 -57.781 -79.740
-57.781  33.065  42.170
-79.740  42.170  63.870

>k1,k2,ii,N(mltr)=k1n,k2n,f1,f2

>f1,f2(out,e)=

      f1
 3.28894e+00  -2.42437e+00  -3.84678e+00

      f2
 0.00000e+00   0.00000e+00   0.00000e+00

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>a:k1,k2(out)=

      k1
 1.56022e+02  -8.05226e+01  -1.31445e+02
-8.05226e+01   4.46216e+01   6.68516e+01
-1.31445e+02   6.68516e+01   1.18129e+02

      k2
 2.01401e+02  -1.03781e+02  -1.59092e+02
-1.03781e+02   5.55282e+01   8.09325e+01
-1.59092e+02   8.09325e+01   1.30365e+02
```

```
>k1,k2,ii,N(mltr)=k1n,k2n,f1,f2

>f1,f2(out,e)=

      f1
2.86026e+00   -2.18331e+00   -3.49269e+00

      f2
0.00000e+00    0.00000e+00    0.00000e+00

>k1n,k2n(mcp)=k1,k2

>ii,one(-)=ii

>ii(if)=a

>:

>k1,k2,ii,N(mltr)=k1n,k2n,f1,f2

>f1,f2(out,e)=

      f1
-4.35703e-01   -4.13821e-01   -9.23808e-01

      f2
3.93805e+00   -2.10267e+00   -3.12784e+00
```

CHAPTER 5

CONCLUSIONS

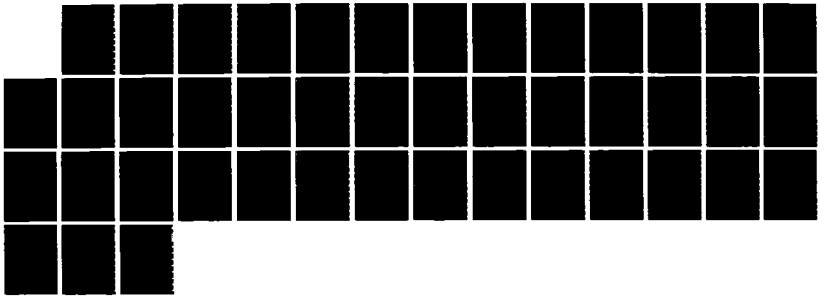
This dissertation studies the computational aspects of iterating two coupled discrete-time Riccati equations. These equations arise as a result of solving LQ descriptor Nash games. The presence of coupling in the Riccati equations complicates the iteration process. This work devises a method which removes the coupling in a numerically robust manner. Then algorithms are engineered that compute the quantities needed to iterate the Riccati equations. Every opportunity is taken to exploit the properties of matrices (e.g., positive-definiteness) that enter into the calculations so as to obtain a savings in computation. The algorithms are coded and the coupled Riccati software is integrated into the L-A-S CACSD language. The novelty of this work is two-fold. First, a new problem is formulated, solved and the solution procedure is implemented as computer code. Second, numerical theory and software are combined under the heading of CACSD to yield a package that allows others to solve single and coupled Riccati equations.

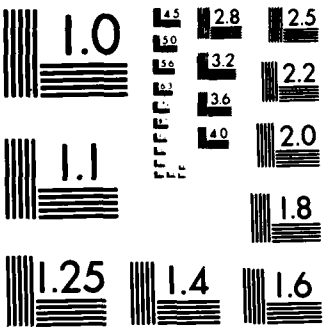
The numerical issues associated with iterating coupled discrete-time Riccati equations are the key focus of this thesis. The software developed for the iteration task is coded in FORTRAN and makes extensive use of the LINPACK library. A structured programming approach has produced low-level algorithms that are very modular and extremely efficient. The final result is a set of six new L-A-S operators that are collectively capable of iterating single and coupled discrete-time Riccati equations.

Although the main contribution of this thesis is the software engineering of the coupled Riccati problem, there are several theoretical advancements which add breadth to the work. The most important of these are the existence and convergence theorems which define the iteration behavior for both finite-horizon and infinite-horizon problems. Of less relevance but not significance is the development of descriptor-variable dynamic games. In addition to theoretical extensions, there are physical and numerical advantages associated with descriptor game formulations. Multirate LQ Nash games were heretofore unposed.

AD-A171 842 ALGORITHMS AND SOFTWARE FOR SOLVING COUPLED DISCRETE-TIME RICCATI EQUATIO..(U) ILLINOIS UNIV AT URBANA DECISION AND CONTROL LAB P J WEST AUG 86 DC-88 2/2

UNCLASSIFIED N00014-84-C-0149 F/G 12/1 NL





There are several directions for future research. For example, the assumption that E is nonsingular could be relaxed. Then uniqueness of the Riccati iterates is lost. Less restrictive conditions insuring convergence for infinite-horizon problems could be sought. A contraction mapping argument could be attempted for the case of unstable plants. Lyapunov-type stability results could be applied to the coupled Riccati maps. A completely different research topic would be the determination of Leader-Follower strategies for descriptor games.

APPENDIX A

SOFTWARE LISTINGS

This appendix contains the FORTRAN computer codes used for a selected number of the low-level, coupled Riccati algorithms. In alphabetical order, the algorithms are MLTPLY, PSICOM and QDFORM.

Algorithm MLTPLY

```
C     ***************************
C     **     M L T P L Y       **
C     ***************************
C     **
C     **  Given the double precision matrices A and B, this subroutine
C     **  computes and returns the variable C defined as the product :
C     **
C     **  C  =  A * B , if the logical variable ATFLAG is False, or
C     **
C     **          T
C     **  C  =  A  * B, if the logical variable ATFLAG is True.
C     **
C     **  where A is an NxM arbitrary matrix (for ATFLAG = False),
C     **            is an MxN arbitrary matrix (for ATFLAG = True),
C     **        B is an MxP arbitrary matrix,
C     **    and C is the resulting NxP matrix.
C     **
      SUBROUTINE   MLTPLY (A,B,C,N,M,P,ATFLAG)
C     **
      DOUBLE PRECISION      A (1)  ,  B (1)  ,  C (1)
      INTEGER               N  ,  M  ,  P
      LOGICAL               ATFLAG
C     **
      INTEGER               I  ,  I1  ,  I2  ,  I3  ,  J  ,  K
C     **
C     **  Branch depending on the value of ATFLAG.
C     **
      IF      (ATFLAG)  GO TO  2
C     **
C     **  Compute A * B and store in C.
C     **
      DO 1   J=1,P
      I3      = (J-1)*N
      DO 1   I=1,N
      I1      = I - N
      I2      = (J-1)*M
      I3      = I3 + 1
      C (I3) = 0.0
```

```
C     **
        DO 1  K=1,M
        I1      = I1 + N
        I2      = I2 + 1
        C (I3) = C (I3)  +  A (I1) * B (I2)
1       CONTINUE
        RETURN
C     **
C     **              T
C     **  Compute A   * B and store in C.
C     **
2       CONTINUE
        DO 3  J=1,P
        I3      = (J-1)*N
        DO 3  I=1,N
        I1      = (I-1)*M
        I2      = (J-1)*M
        I3      = I3 + 1
        C (I3) = 0.0
C     **
        DO 3  K=1,M
        I1      = I1 + 1
        I2      = I2 + 1
        C (I3) = C (I3)  +  A (I1) * B (I2)
3       CONTINUE
        RETURN
        END
```

Algorithm PSICOM

```
C     ***************************
C     **     P S I C O M       **
C     ***************************
C     **
C     **  Given B, KK and R, this subroutine computes and returns
C     **  the variable named PSI defined as :
C     **
C     **                    -1     T
C     **  PSI    =    GAMMA    * B   * KK
C     **
C     **                              T
C     **  where GAMMA   =   R   +   B   * KK * B
C     **        B is a NxP matrix,
C     **        KK is a NxN, positive-semidefinite matrix,
C     **    and R is a PxP, positive matrix.
C     **
C     **  Note : It is assumed that P is less than or equal to N!
C     **
C     **  Several other quantities are calculated and returned for
C     **  possible later use.  These quantities include :
C     **           T
C     **     W = B   * KK ,   GAMMA , XK, and XG where XG is the Cholesky
C     **                                                  T
C     **     factor of GAMMA.  That is, GAMMA = XG   * XG.
C     **     Likewise XK is the Cholesky factor of KK.
C     **
      SUBROUTINE   PSICOM (B,GAMMA,KK,R,W,XG,XK,N,P,PSI)
C     **
C     **  GAMMA contains enough room for 1 double precision PxP matrix.
C     **  W contains enough room for 1 double precision NxN matrix.
C     **  XG contains enough room for 1 double precision PxP matrix.
C     **  XK contains enough room for 1 double precision NxN matrix.
C     **
      DOUBLE PRECISION     B (1)  ,  GAMMA (1)  ,  KK (1)  ,  R (1)
      DOUBLE PRECISION     W (1)  ,  XG (1)  ,  XK (1)  ,  PSI (1)
      INTEGER              N  ,  P
C     **
      INTEGER              I , II , J , K , L , USER
C     **
      DATA                 USER  /6/
C     **
C     **           T
C     **  Compute B   * KK and store in W using algorithm MLTPLY.
C     **
      CALL   MLTPLY (B,KK,W,P,N,N,.TRUE.)
C     **
C     **                              T
C     **  Compute GAMMA  =  R  +  B   * KK * B using algorithm QDFORM.
C     **  Note : Since XG is only computed later, it is used here as a
```

```
C     **          dummy variable for temporary storage by QDFORM.
C     **          Also, CHOFLG = False, CFLAG = True, and ADDFLG = True.
C     **
      CALL    QDFORM (KK,B,R,GAMMA,XK,XG,N,P,.FALSE.,.TRUE., TRUE.)
C     **
C     **                          T
C     **  Factor GAMMA  =  XG  * XG  where XG is upper triangular.
C     **
      K       = P * P
      DO 3   I=1,K
      XG (I) = GAMMA (I)
3     CONTINUE
C     **
      CALL    DPOFA  (XG,P,P,I)
      IF      (I .EQ. 0)  GO TO  5
      WRITE  (USER,4)
4     FORMAT (/' ERROR : PSICOM - GAMMA is not positive definite!')
      RETURN
C     **
C     **  Compute PSI by solving the set of linear equations :
C     **                        T
C     **  GAMMA * PSI  =  B  * K  =  W.
C     **
5     CONTINUE
      I1      = -P
      DO 7   J=1,N
C     **  Copy Jth column of W to Jth column of PSI.
      I1      = I1 + P
      L       = I1
      DO 6   K=1,P
      L       = L + 1
      PSI (L)  =  W (L)
6     CONTINUE
C     **  Compute Jth column of PSI.
      CALL    DPOSL  (XG,P,P,PSI(I1+1))
7     CONTINUE
      RETURN
      END
```

Algorithm QDFORM

```
C     ***************************
C     **     Q D F O R M       **
C     ***************************
C     **
C     **  Given A, B and optionally C, this subroutine computes and
C     **  returns one of the following quadratic forms :
C     **
C     **                    T
C     **  D   =            B  * A * B
C     **
C     **                    T
C     **  D   =   C  +     B  * A * B
C     **
C     **                    T
C     **  D   =   C  -     B  * A * B
C     **
C     **  where A is a NxN, (positive-definite) matrix,
C     **        B is a NxP arbitrary matrix,
C     **    and C is a PxP symmetric matrix.
C     **
C     **  It is assumed that P is less than or equal to N!
C     **
      SUBROUTINE   QDFORM (A,B,C,D,X,Y,N,P,CHOFLG,CFLAG,ADDFLG)
C     **
C     **  Note that the arrays X and Y must be provided by the calling
C     **  routine as temporary storage for the calculation.  Each
C     **  must contain enough room for 1 double precision NxN matrix.
C     **
C     **  This routine has three flags which govern the calculation of a
C     **  quadratic form.
C     **    If CHOFLG is True, then the Cholesky factor of A is already
C     **    available and has been passed in X.  Consequently, the call
C     **    to LINPACK routine DPOFA is skipped.  If CHOFLG is False,
C     **    then the Cholesky factor of A is assumed to be unavailable.
C     **
C     **    If CFLAG is True, then the matrix C is to be included in the
C     **    initialization statement of the matrix multiply DO-Loop.
C     **    If CFLAG is False, then C is never referenced.
C     **
C     **    Assuming CFLAG = True, then ADDFLG is consulted to determine
C     **    if an addition or subtraction is to be performed in the
C     **    matrix multiply DO-Loop.  ADDFLG = True means
C     **           T
C     **    C  +  B  * A * B is computed.  ADDFLG = False means
C     **           T
C     **    C  -  B  * A * B is computed.  If CFLAG = False, then
C     **    ADDFLG is never referenced.
C     **
      DOUBLE PRECISION     A (1)  ,  B (1)  ,  C (1)  ,  D (1)
```

```
      DOUBLE PRECISION      X (1)  ,  Y (1)
      INTEGER               N  ,  P
      LOGICAL               CHOFLG  ,  CFLAG  ,  ADDFLG
C     **
      INTEGER               I  , I1  , I2  , I3  , I4  , J  , K  , USER
C     **
      DATA                  USER  /6/
C     **
C     **  Skip the Cholesky factorization of A if already available.
C     **
      IF     (CHOFLG)  GO TO  3
C     **
C     **                    T
C     **  Factor A  =  X  * X  where X is upper triangular.
C     **
      K      = N * N
      DO 1  I=1,K
      X (I)  = A (I)
1     CONTINUE
C     **
      CALL   DPOFA  (X,N,N,I)
      IF     (I .EQ. 0)  GO TO  3
      WRITE  (USER,2)
2     FORMAT (/' ERROR : QDFORM - Array is not positive definite!')
      RETURN
C     **
C     **  Compute X * B and store in Y taking advantage of the fact
C     **  that X is upper triangular.
C     **
3     CONTINUE
      DO 4  I=1,N
      DO 4  J=1,P
      I3     = (J-1)*N  +  I
      Y (I3) = 0.0
      DO 4  K=I,N
      I1     = (K-1)*N  +  I
      I2     = (J-1)*N  +  K
      Y (I3) = Y (I3)  +  X (I1) * B (I2)
4     CONTINUE
C     **
C     **  Branch if matrix C is referenced.
C     **
      IF     (CFLAG)  GO TO  7
C     **
C     **           T
C     **  Compute Y  * Y and store in D taking advantage of the
C     **  symmetry of D.
C     **
      DO 6  I=1,P
      DO 6  J=I,P
      I1     = (I-1)*N
      I2     = (J-1)*N
      I3     = (J-1)*P  +  I
```

```
      D (I3) = 0.0
      DO 5  K=1,N
      D (I3) = D (I3)  +  Y (I1+K) * Y (I2+K)
5     CONTINUE
      I4     = (I-1)*P  +  J
      D (I4) = D (I3)
6     CONTINUE
      RETURN
C     **
C     **  Branch depending on the value of ADDFLG.
C     **
7     CONTINUE
      IF     (ADDFLG)  GO TO  10
C     **
C     **                    T
C     **  Compute C  -  Y  * Y and store in D taking advantage of
C     **  the symmetry of D.
C     **
      DO 9  I=1,P
      DO 9  J=I,P
      I1     = (I-1)*N
      I2     = (J-1)*N
      I3     = (J-1)*P  +  I
      D (I3) = C (I3)
      DO 8  K=1,N
      D (I3) = D (I3)  -  Y (I1+K) * Y (I2+K)
8     CONTINUE
      I4     = (I-1)*P  +  J
      D (I4) = D (I3)
9     CONTINUE
      RETURN
C     **
C     **                    T
C     **  Compute C  +  Y  * Y and store in D taking advantage of
C     **  the symmetry of D.
C     **
10    CONTINUE
      DO 12  I=1,P
      DO 12  J=I,P
      I1     = (I-1)*N
      I2     = (J-1)*N
      I3     = (J-1)*P  +  I
      D (I3) = C (I3)
      DO 11  K=1,N
      D (I3) = D (I3)  +  Y (I1+K) * Y (I2+K)
11    CONTINUE
      I4     = (I-1)*P  +  J
      D (I4) = D (I3)
12    CONTINUE
      RETURN
      END
```

APPENDIX B

CONTRACTION MAPPING RESULTS

Many smaller results were used in proving the existence of a region where coupled discrete-time Riccati equations constitute a contraction mapping. This appendix is a collection of all the minor results needed for the derivation. In the beginning of this appendix are five lemmas. Each subsequent lemma builds upon the results of previous lemmas. Next, Theorem 3.2, which was stated in Subsection 3.4.2, is proved. Then, Facts 3.2 and 3.3 (also found in Subsection 3.4.2) are proved. The end of the appendix contains a summary page which serves as a quick reference to key results. For ease of notation the 2-subscript on the Euclidean norm, $\|\cdot\|_2$, will be omitted in all the lemmas.

Lemma 1 : Given $R_i = \frac{1}{\epsilon_i}\mathbf{I}$, $\epsilon_i > 0$, i=1,2 then $\| (\Gamma_i(X))^{-1} - (\Gamma_i(Y))^{-1} \| \leq (\epsilon_i)^2 \| B_i \|^2 \cdot \| X - Y \|$

where X and Y are any two positive-semidefinite symmetric matrices and $\Gamma_i(X) \stackrel{\Delta}{=} R_i + B_i^T X B_i$.

Proof : It is straightforward to calculate :

$$
\begin{aligned}
(\Gamma_i(X))^{-1} - (\Gamma_i(Y))^{-1} &= (R_i + B_i^T X B_i)^{-1} - (R_i + B_i^T Y B_i)^{-1} \\
&= (\mathbf{I} + R_i^{-1} B_i^T X B_i)^{-1} R_i^{-1} - R_i^{-1} (\mathbf{I} + B_i^T Y B_i R_i^{-1})^{-1} \\
&= (\mathbf{I} + R_i^{-1} B_i^T X B_i)^{-1} \left| R_i^{-1} - (\mathbf{I} + R_i^{-1} B_i^T X B_i) R_i^{-1} (\mathbf{I} + B_i^T Y B_i R_i^{-1})^{-1} \right| \\
&= (\mathbf{I} + R_i^{-1} B_i^T X B_i)^{-1} \left| R_i^{-1} (\mathbf{I} + B_i^T Y B_i R_i^{-1}) - (\mathbf{I} + R_i^{-1} B_i^T X B_i) R_i^{-1} \right| (\mathbf{I} + B_i^T Y B_i R_i^{-1})^{-1} \\
&= (\mathbf{I} + R_i^{-1} B_i^T X B_i)^{-1} \left| R_i^{-1} B_i^T Y B_i R_i^{-1} - R_i^{-1} B_i^T X B_i R_i^{-1} \right| (\mathbf{I} + B_i^T Y B_i R_i^{-1})^{-1} \\
&= (\mathbf{I} + R_i^{-1} B_i^T X B_i)^{-1} R_i^{-1} B_i^T \left| Y - X \right| B_i R_i^{-1} (\mathbf{I} + B_i^T Y B_i R_i^{-1})^{-1} . \qquad \text{(B-1.1)}
\end{aligned}
$$

At this juncture, let $R_i = \frac{1}{\epsilon_i}\mathbf{I}$ which implies that $(R_i)^{-1} = \epsilon_i \mathbf{I}$. Then, from (B-1.1) it follows that

$$\| (\Gamma_i(X))^{-1} - (\Gamma_i(Y))^{-1} \| = \| (I + \epsilon_i B_i^T X B_i)^{-1} \epsilon_i B_i^T \left| Y - X \right| B_i \epsilon_i (I + B_i^T Y B_i \epsilon_i)^{-1} \|$$

$$= (\epsilon_i)^2 \| (I + \epsilon_i B_i^T X B_i)^{-1} B_i^T \left| Y - X \right| B_i (I + \epsilon_i B_i^T Y B_i)^{-1} \|$$

$$\leq (\epsilon_i)^2 \| (I + \epsilon_i B_i^T X B_i)^{-1} \| \cdot \| B_i \|^2 \cdot \| Y - X \| \cdot \| (I + \epsilon_i B_i^T Y B_i)^{-1} \| . \qquad \text{(B-1.2)}$$

Since, $\| (I + \epsilon_i B_i^T X B_i)^{-1} \| = \dfrac{1}{\underline{\sigma}(I + \epsilon_i B_i^T X B_i)}$ and both I and $\epsilon_i B_i^T X B_i$ are positive-semidefinite matrices, then $\underline{\sigma}(I + \epsilon_i B_i^T X B_i) \geq \underline{\sigma}(I) + \underline{\sigma}(\epsilon_i B_i^T X B_i) \geq \underline{\sigma}(I) = 1$. Hence,

$$\| (I + \epsilon_i B_i^T X B_i)^{-1} \| = \frac{1}{\underline{\sigma}(I + \epsilon_i B_i^T X B_i)} \leq 1. \qquad \text{(B-1.3)}$$

A similar argument concludes that

$$\| (I + \epsilon_i B_i^T Y B_i)^{-1} \| = \frac{1}{\underline{\sigma}(I + \epsilon_i B_i^T Y B_i)} \leq 1. \qquad \text{(B-1.4)}$$

Finally, since $\| Y - X \| = \| X - Y \|$, we have from (B-1.2) that

$$\| (\Gamma_i(X))^{-1} - (\Gamma_i(Y))^{-1} \| \leq (\epsilon_i)^2 \| B_i \|^2 \cdot \| X - Y \| . \qquad \text{(B-1.5)}$$

□

Lemma 2 : Given $R_i = \frac{1}{\epsilon_i} I$, $\epsilon_i > 0$, i=1,2 then

$$\| \Psi_i(X) - \Psi_i(Y) \| \leq \epsilon_i \| B_i \| \cdot \left| 1 + \frac{\epsilon_i}{2} \| B_i \|^2 \cdot \| X + Y \| \right| \cdot \| X - Y \| \qquad \text{(B-2.1)}$$

where X and Y are any two positive-semidefinite symmetric matrices and $\Psi_i(X) \stackrel{\Delta}{=} (\Gamma_i(X))^{-1} B_i^T X$. Furthermore, if X and Y are confined to a closed ball of radius δ_i then the following bound is obtained :

$$\| \Psi_i(X) - \Psi_i(Y) \| \leq \epsilon_i \| B_i \| \cdot \left| 1 + \epsilon_i \delta_i \| B_i \|^2 \right| \cdot \| X - Y \| \, . \qquad \text{(B-2.2)}$$

Proof : For this and subsequent proofs we will utilize the matrix identity :

$$W X - Y Z = \frac{1}{2} \cdot \left| (W - Y)(X + Z) + (W + Y)(X - Z) \right| \qquad \text{(B-2.3)}$$

where W , X , Y , and Z are arbitrary but compatibly dimensioned matrices. Now, notice that

$$\| (\Gamma_i(X))^{-1} \| = \frac{1}{\underline{\sigma}(R_i + B_i^T X B_i)} \leq \frac{1}{\underline{\sigma}(R_i) + \underline{\sigma}(B_i^T X B_i)} \leq \frac{1}{\underline{\sigma}(R_i)} = \epsilon_i \, . \qquad \text{(B-2.4)}$$

Use (B-2.3) and the triangle inequality to compute

$$\begin{aligned} &\| \Psi_i(X) - \Psi_i(Y) \| \\ &= \frac{1}{2} \| \left| (\Gamma_i(X))^{-1} - (\Gamma_i(Y))^{-1} \right| B_i (X + Y) + \left| (\Gamma_i(X))^{-1} + (\Gamma_i(Y))^{-1} \right| B_i (X - Y) \| \\ &\leq \frac{\| B_i \|}{2} \cdot \left| \| (\Gamma_i(X))^{-1} - (\Gamma_i(Y))^{-1} \| \cdot \| X + Y \| + \| (\Gamma_i(X))^{-1} + (\Gamma_i(Y))^{-1} \| \cdot \| X - Y \| \right| . \end{aligned}$$

An additional application of the triangle inequality coupled with the result of Lemma 1 and (B-2.4) yields :

$$\begin{aligned} &\| \Psi_i(X) - \Psi_i(Y) \| \\ &\leq \frac{\| B_i \|}{2} \cdot \left| (\epsilon_i)^2 \| B_i \|^2 \cdot \| X + Y \| \cdot \| X - Y \| + 2 \epsilon_i \| X - Y \| \right| \\ &= \epsilon_i \| B_i \| \cdot \left| 1 + \frac{\epsilon_i}{2} \| B_i \|^2 \cdot \| X + Y \| \right| \cdot \| X - Y \| \, . \end{aligned}$$

Hence, (B-2.1) is verified. Finally, if X and Y are contained in a ball of radius δ_i , then $\| X \| \leq \delta_i$ and $\| Y \| \leq \delta_i$. Thus,

$$
\begin{aligned}
\| \Psi_i(X) - \Psi_i(Y) \| &\leq \epsilon_i \| B_i \| \cdot \left| 1 + \frac{\epsilon_i}{2} \| B_i \|^2 \cdot \| X + Y \| \right| \| X - Y \| \\
&\leq \epsilon_i \| B_i \| \cdot \left| 1 + \frac{\epsilon_i}{2} \| B_i \|^2 \cdot \left(\| X \| + \| Y \| \right) \right| \cdot \| X - Y \| \\
&\leq \epsilon_i \| B_i \| \cdot \left| 1 + \frac{\epsilon_i}{2} \| B_i \|^2 \cdot \left(\delta_i + \delta_i \right) \right| \cdot \| X - Y \| \\
&= \epsilon_i \| B_i \| \cdot \left| 1 + \epsilon_i \delta_i \| B_i \|^2 \right| \cdot \| X - Y \| .
\end{aligned}
$$

□

Lemma 3 : Given $R_i = \frac{1}{\epsilon_i}\mathbf{I}$, $\epsilon_i > 0$, i=1,2 and any two positive-semidefinite symmetric matrices X_1 and X_2, then whenever $\epsilon_i < \frac{1}{\|B_i\|^2 \cdot \|X_i\|}$ holds simultaneously for i=1,2, it follows that

$$\left\| \left[\Xi_i(X_1, X_2)\right]^{-1} \right\| \leq \frac{1}{\left[1 - \epsilon_1\epsilon_2 \|B_1\|^2 \cdot \|B_2\|^2 \cdot \|X_1\| \cdot \|X_2\|\right]}$$

$$= \frac{1}{1 - \lambda_1(X_1)\lambda_2(X_2)} \qquad \text{(B-3.1)}$$

where

$$\Xi_1(X_1, X_2) \triangleq \mathbf{I} - \Psi_1(X_1)B_2\Psi_2(X_2)B_1 . \qquad \text{(B-3.2a)}$$

$$\Xi_2(X_1, X_2) \triangleq \mathbf{I} - \Psi_2(X_2)B_1\Psi_1(X_1)B_2 . \quad \text{and} \qquad \text{(B-3.2b)}$$

$$\lambda_i(X) \triangleq \epsilon_i \|B_i\|^2 \cdot \|X\| . \qquad \text{(B-3.3)}$$

Furthermore, if X_i is confined to a closed ball of radius δ_i, i=1,2, then whenever $\epsilon_i < \frac{1}{\sqrt{2}\,\delta_i \|B_i\|^2}$ the following bound is obtained :

$$\left\| \left[\Xi_i(X_1, X_2)\right]^{-1} \right\| \leq 2 . \qquad \text{(B-3.4)}$$

Proof : For this proof, we use (B-2.4) of Lemma 2 and the definition of $\Psi_i(X)$ to conclude that

$$\|\Psi_i(X)\| \leq \epsilon_i \|B_i\| \cdot \|X\| . \qquad \text{(B-3.5)}$$

Define

$$\Omega_1(X_1, X_2) \triangleq \Psi_1(X_1)B_2\Psi_2(X_2)B_1 . \quad \text{and} \qquad \text{(B-3.6a)}$$

$$\Omega_2(X_1, X_2) \triangleq \Psi_2(X_2)B_1\Psi_1(X_1)B_2 . \qquad \text{(B-3.6b)}$$

Then, $\left\| \left[\Xi_i(X_1, X_2)\right]^{-1} \right\| = \left\| \left[\mathbf{I} - \Omega_i(X_1, X_2)\right]^{-1} \right\|$. Note that if $\|\Omega_i(X_1, X_2)\| < 1$, then $\underline{\sigma}(\mathbf{I} - \Omega_i(X_1, X_2)) \geq \underline{\sigma}(\mathbf{I}) - \overline{\sigma}(\Omega_i(X_1, X_2))$. Suppose that $\|\Omega_i(X_1, X_2)\| < 1$. Hence,

$$\left\| \left[\Xi_i(X_1, X_2)\right]^{-1} \right\| = \left\| \left[\mathbf{I} - \Omega_i(X_1, X_2)\right]^{-1} \right\|$$

$$= \frac{1}{\underline{\sigma}(\mathbf{I} - \Omega_i(X_1, X_2))}$$

$$\leqslant \frac{1}{\underline{\sigma}(I) - \overline{\sigma}(\Omega_i(X_1, X_2))}$$

$$= \frac{1}{1 - \| \Omega_i(X_1, X_2) \|} \quad . \tag{B-3.7}$$

Next, for both i=1 and i=2 :

$$\| \Omega_i(X_1, X_2) \| \leqslant \| B_1 \| \cdot \| B_2 \| \cdot \| \Psi_1(X_1) \| \cdot \| \Psi_2(X_2) \|$$

$$\leqslant \| B_1 \| \cdot \| B_2 \| \cdot \left(\epsilon_1 \| B_1 \| \cdot \| X_1 \| \right) \cdot \left(\epsilon_2 \| B_2 \| \cdot \| X_2 \| \right)$$

$$= \epsilon_1 \epsilon_2 \| B_1 \|^2 \cdot \| B_2 \|^2 \cdot \| X_1 \| \cdot \| X_2 \| \quad . \tag{B-3.8}$$

Define $\overline{\epsilon_i} \triangleq \left(\| B_i \|^2 \cdot \| X_i \| \right)^{-1}$, i=1,2. So, whenever $\epsilon_i < \overline{\epsilon_i}$ holds simultaneously for all $i \in \{ 1, 2 \}$, then it follows that $\| \Omega_i(X_1, X_2) \| < 1$. Consequently, for all $\epsilon_i \in [0, \overline{\epsilon_i})$, i=1,2

$$\| \left(\Xi_i(X_1, X_2) \right)^{-1} \| \leqslant \frac{1}{1 - \| \Omega_i(X_1, X_2) \|}$$

$$\leqslant \frac{1}{1 - \epsilon_1 \epsilon_2 \| B_1 \|^2 \cdot \| B_2 \|^2 \cdot \| X_1 \| \cdot \| X_2 \|} \quad .$$

In view of (B-3.3) ,

$$\frac{1}{1 - \lambda_1(X_1) \lambda_2(X_2)} = \frac{1}{1 - \epsilon_1 \epsilon_2 \| B_1 \|^2 \cdot \| B_2 \|^2 \cdot \| X_1 \| \cdot \| X_2 \|} \quad . \tag{B-3.9}$$

Therefore, (B-3.1) is proved. Finally, take $\overline{\epsilon_i} = \frac{1}{\sqrt{2}\, \delta_i \| B_i \|^2}$. Then, $\lambda_1(X_1) \lambda_2(X_2) \leqslant \frac{1}{2}$. Thus,

$$\| (\Xi_i(X_1, X_2))^{-1} \| \leqslant \frac{1}{1 - \frac{1}{2}} = 2 \quad .$$

□

Lemma 4 : Given $R_i = \frac{1}{\epsilon_i} I$, $\epsilon_i > 0$, i=1,2 and any four positive-semidefinite symmetric matrices X_1, X_2, Y_1, and Y_2, such that $\| X_i \| \leq \delta_i$ and $\| Y_i \| \leq \delta_i$, then if $0 < \epsilon_i < \bar{\epsilon}_i \triangleq \frac{1}{\delta_i \| B_i \|^2}$ holds simultaneously for all $i \in \{1,2\}$, it follows that :

$$\left. \begin{aligned} & \| \left[\Xi_i(X_1, X_2)\right]^{-1} - \left[\Xi_i(Y_1, Y_2)\right]^{-1} \| \\ & \leq \frac{\nu_1 \nu_2 (1+\nu_1)}{\delta_1 (1-\nu_1 \nu_2)^2} \cdot \| X_1 - Y_1 \| + \frac{\nu_1 \nu_2 (1+\nu_2)}{\delta_2 (1-\nu_1 \nu_2)^2} \cdot \| X_2 - Y_2 \| \end{aligned} \right| \qquad \text{(B-4.1)}$$

where $\nu_i \triangleq \epsilon_i \delta_i \| B_i \|^2 < 1$.

Proof : Using notation of Lemma 3, (B-3.6a-b) :

$$\Xi_i(X_1, X_2) = I - \Omega_i(X_1, X_2) .$$

Then via a development that is similar to Lemma 1, (B-1.1), we obtain

$$\left[\Xi_i(X_1, X_2)\right]^{-1} - \left[\Xi_i(Y_1, Y_2)\right]^{-1} = \left[\Xi_i(X_1, X_2)\right]^{-1} \left| \Omega_i(X_1, X_2) - \Omega_i(Y_1, Y_2) \right| \left[\Xi_i(Y_1, Y_2)\right]^{-1} .$$

Now, for i=1

$$\begin{aligned} \Omega_1(X_1, X_2) - \Omega_1(Y_1, Y_2) &= \Psi_1(X_1) B_2 \Psi_2(X_2) B_1 - \Psi_1(Y_1) B_2 \Psi_2(Y_2) B_1 \\ &= \frac{1}{2} \left| \begin{matrix} (\Psi_1(X_1) - \Psi_1(Y_1)) B_2 (\Psi_2(X_2) + \Psi_2(Y_2)) B_1 + \\ (\Psi_1(X_1) + \Psi_1(Y_1)) B_2 (\Psi_2(X_2) - \Psi_2(Y_2)) B_1 \end{matrix} \right| . \end{aligned}$$

Likewise,

$$\Omega_2(X_1, X_2) - \Omega_2(Y_1, Y_2) = \frac{1}{2} \left| \begin{matrix} (\Psi_2(X_2) - \Psi_2(Y_2)) B_1 (\Psi_1(X_1) + \Psi_1(Y_1)) B_2 + \\ (\Psi_2(X_2) + \Psi_2(Y_2)) B_1 (\Psi_1(X_1) - \Psi_1(Y_1)) B_2 \end{matrix} \right| .$$

Remark : $\Psi_1(\cdot)$ is *always* a function of the first variable and $\Psi_2(\cdot)$ is *always* a function of the second variable.

Making use of the triangle inequality, it is determined that

$$\| \Omega_i(X_1, X_2) - \Omega_i(Y_1, Y_2) \| \leq \frac{1}{2} \| B_1 \| \cdot \| B_2 \| \cdot \left| \begin{matrix} \| \Psi_1(X_1) - \Psi_1(Y_1) \| \cdot \| \Psi_2(X_2) + \Psi_2(Y_2) \| + \\ \| \Psi_1(X_1) + \Psi_1(Y_1) \| \cdot \| \Psi_2(X_2) - \Psi_2(Y_2) \| \end{matrix} \right| .$$

Furthermore,

$$\| \Psi_i(X_i) + \Psi_i(Y_i) \| \leq \epsilon_i \| B_i \| \left(\| X_i \| + \| Y_i \| \right) \leq 2 \epsilon_i \delta_i \| B_i \| . \tag{B-4.2}$$

Also,

$$\begin{aligned} \| \Psi_i(X_i) - \Psi_i(Y_i) \| &\leq \epsilon_i \| B_i \| \cdot \left| 1 + \frac{\epsilon_i}{2} \| B_i \|^2 \cdot \| X_i + Y_i \| \right| \cdot \| X_i - Y_i \| \\ &\leq \epsilon_i \| B_i \| \cdot \left| 1 + \epsilon_i \delta_i \| B_i \|^2 \right| \cdot \| X_i - Y_i \| \\ &= \epsilon_i \| B_i \| \cdot \left| 1 + \nu_i \right| \cdot \| X_i - Y_i \| . \end{aligned} \tag{B-4.3}$$

Therefore, putting all these facts together yields the following bound :

$$\begin{aligned} \| \Omega_i(X_1, X_2) - \Omega_i(Y_1, Y_2) \| &\leq \epsilon_1 \epsilon_2 \| B_1 \|^2 \cdot \| B_2 \|^2 \cdot \left| \begin{matrix} \delta_2 (1 + \nu_1) \| X_1 - Y_1 \| + \\ \delta_1 (1 + \nu_2) \| X_2 - Y_2 \| \end{matrix} \right| \\ &= \left| \begin{matrix} \epsilon_1 \nu_2 (1 + \nu_1) \| B_1 \|^2 \cdot \| X_1 - Y_1 \| + \\ \epsilon_2 \nu_1 (1 + \nu_2) \| B_2 \|^2 \cdot \| X_2 - Y_2 \| \end{matrix} \right| . \end{aligned} \tag{B-4.4}$$

Note : The derivation of this upper bound for $\| \Omega_i(X_1, X_2) - \Omega_i(Y_1, Y_2) \|$ does not impose any restrictions on the magnitude of ϵ_1 or ϵ_2! However, in order to bound $\left\| \left[\Xi_i(X_1, X_2) \right]^{-1} - \left[\Xi_i(Y_1, Y_2) \right]^{-1} \right\|$, we must now invoke the definition of $\overline{\epsilon_i} \triangleq \frac{1}{\delta_i \| B_i \|^2}$. Hence, for all $0 < \epsilon_i < \overline{\epsilon_i}$, the hypothesis of Lemma 3 is satisfied and we conclude that

$$\begin{aligned} \left\| \left[\Xi_i(X_1, X_2) \right]^{-1} \right\| &\leq \frac{1}{1 - \lambda_1(X_1) \lambda_2(X_2)} \\ &= \frac{1}{1 - \epsilon_1 \epsilon_2 \| B_1 \|^2 \cdot \| B_2 \|^2 \cdot \| X_1 \| \cdot \| X_2 \|} \\ &\leq \frac{1}{1 - \left(\epsilon_1 \delta_1 \| B_1 \|^2 \right) \cdot \left(\epsilon_2 \delta_2 \| B_2 \|^2 \right)} \end{aligned}$$

$$= \frac{1}{1 - \nu_1 \nu_2} \cdot \tag{B-4.5}$$

Clearly,

$$\| \left[\Xi_i(X_1, X_2)\right]^{-1} - \left[\Xi_i(Y_1, Y_2)\right]^{-1} \|$$

$$\leq \| \left[\Xi_i(X_1, X_2)\right]^{-1} \| \cdot \| \left[\Xi_i(Y_1, Y_2)\right]^{-1} \| \cdot | \Omega_i(X_1, X_2) - \Omega_i(Y_1, Y_2) |$$

$$\leq \frac{| \Omega_i(X_1, X_2) - \Omega_i(Y_1, Y_2) |}{(1 - \nu_1 \nu_2)^2}$$

$$\leq \frac{\epsilon_1 \nu_2 (1 + \nu_1) | B_1 |^2}{(1 - \nu_1 \nu_2)^2} \cdot | X_1 - Y_1 | + \frac{\epsilon_2 \nu_1 (1 + \nu_2) | B_2 |^2}{(1 - \nu_1 \nu_2)^2} \cdot | X_2 - Y_2 |$$

$$= \frac{\nu_1 \nu_2 (1 + \nu_1)}{\delta_1 (1 - \nu_1 \nu_2)^2} \cdot | X_1 - Y_1 | + \frac{\nu_1 \nu_2 (1 + \nu_2)}{\delta_2 (1 - \nu_1 \nu_2)^2} \cdot | X_2 - Y_2 | .$$

□

Lemma 5 : Given $R_i = \frac{1}{\epsilon_i} I$, $\epsilon_i > 0$, i=1,2 and any two positive-semidefinite symmetric matrices X_1 and X_2, with $\| X_i \| \leq \delta_i$, then whenever $0 < \epsilon_i < \bar{\epsilon}_i \triangleq \frac{1}{\delta_i \| B_i \|^2}$ holds for all $i \in \{1,2\}$, it follows that :

$$\left.\begin{aligned} \| B_1 F_1(X_1, X_2) \| &\leq \frac{\xi \nu_1 (1+\nu_2)}{1-\nu_1 \nu_2} \quad \text{and} \\ \| B_2 F_2(X_1, X_2) \| &\leq \frac{\xi \nu_2 (1+\nu_1)}{1-\nu_1 \nu_2} \end{aligned}\right| \qquad \text{(B-5.1)}$$

where $\xi \triangleq \| A \|$ and $\nu_i \triangleq \epsilon_i \delta_i \| B_i \|^2 < 1$.

Proof : By definition :

$$B_1 F_1(X_1, X_2) = B_1 \left[\Xi_1(X_1, X_2)\right]^{-1} \Psi_1(X_1) \left[I - B_2 \Psi_2(X_2) \right] A \quad , \text{ and}$$

$$B_2 F_2(X_1, X_2) = B_2 \left[\Xi_2(X_1, X_2)\right]^{-1} \Psi_2(X_2) \left[I - B_1 \Psi_1(X_1) \right] A \quad .$$

Thus,

$$\| B_1 F_1(X_1, X_2) \| \leq \xi \| B_1 \| \cdot \| \left[\Xi_1(X_1, X_2)\right]^{-1} \| \cdot \| \Psi_1(X_1) \| \cdot \| I - B_2 \Psi_2(X_2) \|$$

$$\| B_2 F_2(X_1, X_2) \| \leq \xi \| B_2 \| \cdot \| \left[\Xi_2(X_1, X_2)\right]^{-1} \| \cdot \| \Psi_2(X_2) \| \cdot \| I - B_1 \Psi_1(X_1) \|$$

where $\xi \triangleq \| A \|$.

Since, $\| \Psi_i(X) \| \leq \epsilon_i \| B_i \| \cdot \| X \| \leq \epsilon_i \delta_i \| B_i \|$.

$$\begin{aligned} \| B_1 F_1(X_1, X_2) \| &\leq \xi \epsilon_1 \delta_1 \| B_1 \|^2 \cdot \| \left[\Xi_1(X_1, X_2)\right]^{-1} \| \cdot \| I - B_2 \Psi_2(X_2) \| \\ &= \xi \nu_1 \| \left[\Xi_1(X_1, X_2)\right]^{-1} \| \cdot \| I - B_2 \Psi_2(X_2) \| \\ &\leq \frac{\xi \nu_1}{1-\nu_1 \nu_2} \cdot \| I - B_2 \Psi_2(X_2) \| \\ &\leq \frac{\xi \nu_1}{1-\nu_1 \nu_2} \cdot \left[1 + \| B_2 \| \cdot \| \Psi_2(X_2) \| \right] \leq \frac{\xi \nu_1 (1+\nu_2)}{1-\nu_1 \nu_2} \quad . \end{aligned}$$

Similarly,

$$\begin{aligned}
\| B_2 F_2(X_1, X_2) \| &\leq \xi \epsilon_2 \delta_2 \| B_2 \|^2 \cdot \| \left[\Xi_2(X_1, X_2) \right]^{-1} \| \cdot \| I - B_1 \Psi_1(X_1) \| \\
&= \xi \nu_2 \| \left[\Xi_2(X_1, X_2) \right]^{-1} \| \cdot \| I - B_1 \Psi_1(X_1) \| \\
&\leq \frac{\xi \nu_2}{1 - \nu_1 \nu_2} \cdot \| I - B_1 \Psi_1(X_1) \| \\
&\leq \frac{\xi \nu_2}{1 - \nu_1 \nu_2} \cdot \left(1 + \| B_1 \| \cdot \| \Psi_1(X_1) \| \right) \leq \frac{\xi \nu_2 (1 + \nu_1)}{1 - \nu_1 \nu_2} .
\end{aligned}$$

□

Next, the proof of Theorem 3.2 is given. The significance of this result is that it defines Lipschitz constants for the feedback expressions. The proof is streamlined by utilizing the results of the previous lemmas.

Theorem **3.2 :** Lipschitz Constants for the Feedbacks

Given $(X, Y), (Z, W) \in \Delta$, then

$$\| B_1 (F_1(Z, W) - F_1(X, Y)) \|_2 \leq \alpha_{11} \| X - Z \|_2 + \alpha_{12} \| Y - W \|_2$$

$$\| B_2 (F_2(Z, W) - F_2(X, Y)) \|_2 \leq \alpha_{21} \| X - Z \|_2 + \alpha_{22} \| Y - W \|_2$$

where $\alpha_{11} = \dfrac{\xi \bar{\nu}_1 \bar{\kappa}_1 \bar{\kappa}_2}{\bar{\delta}_1}$, $\alpha_{12} = \dfrac{\xi \bar{\nu}_1 \bar{\nu}_2 \bar{\kappa}_1 \bar{\kappa}_2}{\bar{\delta}_2}$, $\alpha_{21} = \dfrac{\xi \bar{\nu}_1 \bar{\nu}_2 \bar{\kappa}_1 \bar{\kappa}_2}{\bar{\delta}_1}$, and $\alpha_{22} = \dfrac{\xi \bar{\nu}_2 \bar{\kappa}_1 \bar{\kappa}_2}{\bar{\delta}_2}$.

Proof **:** Consider the first inequality dropping all 2-subscripts. By definition :

$$\| B_1 (F_1(Z, W) - F_1(X, Y)) \|$$

$$= \left\| B_1 \left[\left(\Xi_1(Z, W) \right)^{-1} \Psi_1(Z) \left(I - B_2 \Psi_2(W) \right) - \left(\Xi_1(X, Y) \right)^{-1} \Psi_1(X) \left(I - B_2 \Psi_2(Y) \right) \right] A \right\|$$

$$\leq \xi \| B_1 \|_2 \cdot \left\| \left(\Xi_1(Z, W) \right)^{-1} \Psi_1(Z) \left(I - B_2 \Psi_2(W) \right) - \left(\Xi_1(X, Y) \right)^{-1} \Psi_1(X) \left(I - B_2 \Psi_2(Y) \right) \right\| .$$

The last term needs further investigation. Using the matrix identity (B-2.3), we can write :

$$\left\| \left(\Xi_1(Z, W) \right)^{-1} \Psi_1(Z) \left(I - B_2 \Psi_2(W) \right) - \left(\Xi_1(X, Y) \right)^{-1} \Psi_1(X) \left(I - B_2 \Psi_2(Y) \right) \right\|$$

$$\leq \frac{\left\| \left[\left(\Xi_1(Z, W) \right)^{-1} \Psi_1(Z) - \left(\Xi_1(X, Y) \right)^{-1} \Psi_1(X) \right] \cdot \left[2I - B_2 (\Psi_2(Y) + \Psi_2(W)) \right] \right\|}{2}$$

$$+ \frac{\left\| \left[\left(\Xi_1(Z, W) \right)^{-1} \Psi_1(Z) + \left(\Xi_1(X, Y) \right)^{-1} \Psi_1(X) \right] \cdot \left[B_2 (\Psi_2(Y) - \Psi_2(W)) \right] \right\|}{2} .$$

Note that from (B-3.5) and the fact that $Y, W \in B_{\bar{\delta}_2}$ by hypothesis :

$$\| I - B_2 \Psi_2(Y) \| \leq 1 + \bar{\nu}_2$$

and similarly

$$\| I - B_2 \Psi_2(W) \| \leq 1 + \bar{\nu}_2 .$$

Therefore,

$$\left\| \left[\Xi_1(Z,W)\right]^{-1}\Psi_1(Z)\left[I - B_2\Psi_2(W)\right] - \left[\Xi_1(X,Y)\right]^{-1}\Psi_1(X)\left[I - B_2\Psi_2(Y)\right] \right\|$$

$$\leq (1+\bar{\nu}_2)\cdot \left\| \left[\Xi_1(Z,W)\right]^{-1}\Psi_1(Z) - \left[\Xi_1(X,Y)\right]^{-1}\Psi_1(X) \right\| +$$

$$\frac{\|B_2\|}{2}\cdot \left\| \left[\Xi_1(Z,W)\right]^{-1}\Psi_1(Z) + \left[\Xi_1(X,Y)\right]^{-1}\Psi_1(X) \right\| \cdot \|\Psi_2(Y) - \Psi_2(W)\|$$

$$\leq \left| \begin{array}{l} (1+\bar{\nu}_2)\cdot \left\| \left[\Xi_1(Z,W)\right]^{-1}\Psi_1(Z) - \left[\Xi_1(X,Y)\right]^{-1}\Psi_1(X) \right\| + \\ \dfrac{\|B_2\|}{2(1-\bar{\nu}_1\bar{\nu}_2)}\cdot \left[\|\Psi_1(Z)\| + \|\Psi_1(X)\|\right]\cdot \|\Psi_2(Y) - \Psi_2(W)\| \end{array} \right. \quad \text{(B-T.3.2.1)}$$

where (B-4.5) has been used. Summarizing the results thus far :

$$\|B_1(F_1(Z,W) - F_1(X,Y))\|$$

$$\leq \xi(1+\bar{\nu}_2)\|B_1\|\cdot \left\| \left[\Xi_1(Z,W)\right]^{-1}\Psi_1(Z) - \left[\Xi_1(X,Y)\right]^{-1}\Psi_1(X) \right\| +$$

$$\frac{\xi\|B_1\|\cdot\|B_2\|}{2(1-\bar{\nu}_1\bar{\nu}_2)}\cdot \left[\|\Psi_1(Z)\| + \|\Psi_1(X)\|\right]\cdot \|\Psi_2(Y) - \Psi_2(W)\|$$

$$\leq \xi(1+\bar{\nu}_2)\|B_1\|\cdot \left\| \left[\Xi_1(Z,W)\right]^{-1}\Psi_1(Z) - \left[\Xi_1(X,Y)\right]^{-1}\Psi_1(X) \right\| +$$

$$\frac{\xi\bar{\nu}_1\|B_2\|}{1-\bar{\nu}_1\bar{\nu}_2}\cdot\|\Psi_2(Y) - \Psi_2(W)\| .$$

Now, an additional use of the matrix identity (B-2.3) on the first term of equation (B-T.3.2.1) yields :

$$\left\| \left[\Xi_1(Z,W)\right]^{-1}\Psi_1(Z) - \left[\Xi_1(X,Y)\right]^{-1}\Psi_1(X) \right\|$$

$$= \frac{1}{2}\left\| \left| \left[\Xi_1(Z,W)\right]^{-1} - \left[\Xi_1(X,Y)\right]^{-1} \right| \cdot \left[\Psi_1(Z) + \Psi_1(X)\right] + \right.$$

$$\left. \left| \left[\Xi_1(Z,W)\right]^{-1} + \left[\Xi_1(X,Y)\right]^{-1} \right| \cdot \left[\Psi_1(Z) - \Psi_1(X)\right] \right\|$$

$$\leq \epsilon_1\bar{\delta}_1\|B_1\|\cdot \left\| \left[\Xi_1(Z,W)\right]^{-1} - \left[\Xi_1(X,Y)\right]^{-1} \right\| + \frac{1}{1-\bar{\nu}_1\bar{\nu}_2}\cdot\|\Psi_1(Z) - \Psi_1(X)\| .$$

Hence,

$$
\begin{aligned}
&\| B_1 (F_1(Z,W) - F_1(X,Y)) \| \\
&\leq \xi \bar{\nu}_1 (1 + \bar{\nu}_2) \cdot \left\| \left[\Xi_1(Z,W) \right]^{-1} - \left[\Xi_1(X,Y) \right]^{-1} \right\| + \\
&\quad \frac{\xi (1 + \bar{\nu}_2) \| B_1 \|}{1 - \bar{\nu}_1 \bar{\nu}_2} \cdot \| \Psi_1(Z) - \Psi_1(X) \| + \frac{\xi \bar{\nu}_1 \| B_2 \|}{1 - \bar{\nu}_1 \bar{\nu}_2} \cdot \| \Psi_2(Y) - \Psi_2(W) \| \\
&\leq \xi \bar{\nu}_1 (1 + \bar{\nu}_2) \left| \frac{\bar{\nu}_1 \bar{\nu}_2}{(1 - \bar{\nu}_1 \bar{\nu}_2)^2} \left(\frac{1 + \bar{\nu}_1}{\bar{\delta}_1} \| X - Z \| + \frac{1 + \bar{\nu}_2}{\bar{\delta}_2} \| Y - W \| \right) \right| + \\
&\quad \frac{\xi (1 + \bar{\nu}_2)}{1 - \bar{\nu}_1 \bar{\nu}_2} \cdot \frac{\bar{\nu}_1 (1 + \bar{\nu}_1)}{\bar{\delta}_1} \cdot \| X - Z \| + \frac{\xi \bar{\nu}_1}{1 - \bar{\nu}_1 \bar{\nu}_2} \cdot \frac{\bar{\nu}_2 (1 + \bar{\nu}_2)}{\bar{\delta}_2} \cdot \| Y - W \| \\
&= \frac{\xi \bar{\nu}_1 (1 + \bar{\nu}_1) (1 + \bar{\nu}_2)}{\bar{\delta}_1 (1 - \bar{\nu}_1 \bar{\nu}_2)} \cdot \left| \frac{\bar{\nu}_1 \bar{\nu}_2}{1 - \bar{\nu}_1 \bar{\nu}_2} + 1 \right| \cdot \| X - Z \| + \\
&\quad \frac{\xi \bar{\nu}_1 \bar{\nu}_2 (1 + \bar{\nu}_2)}{\bar{\delta}_2 (1 - \bar{\nu}_1 \bar{\nu}_2)} \cdot \left| \frac{\bar{\nu}_1 (1 + \bar{\nu}_2)}{1 - \bar{\nu}_1 \bar{\nu}_2} + 1 \right| \cdot \| Y - W \| \\
&= \frac{\xi \bar{\nu}_1 (1 + \bar{\nu}_1) (1 + \bar{\nu}_2)}{\bar{\delta}_1 (1 - \bar{\nu}_1 \bar{\nu}_2)^2} \cdot \| X - Z \| + \frac{\xi \bar{\nu}_1 \bar{\nu}_2 (1 + \bar{\nu}_1) (1 + \bar{\nu}_2)}{\bar{\delta}_2 (1 - \bar{\nu}_1 \bar{\nu}_2)^2} \cdot \| Y - W \| \\
&= \frac{\xi \bar{\nu}_1 \bar{\kappa}_1 \bar{\kappa}_2}{\bar{\delta}_1} \cdot \| X - Z \| + \frac{\xi \bar{\nu}_1 \bar{\nu}_2 \bar{\kappa}_1 \bar{\kappa}_2}{\bar{\delta}_2} \cdot \| Y - W \| \\
&\stackrel{\Delta}{=} \alpha_{11} \| X - Z \| + \alpha_{12} \| Y - W \| .
\end{aligned}
$$

A completely analogous argument can be developed for the second inequality which results in the appropriate definitions for α_{21} and α_{22}.

□

Now, the proofs of Facts 3.2 and 3.3 are stated.

***Fact* 3.2 :** Given any $X, Y \in B_{\delta_i}$ and $R_i \triangleq \frac{1}{\epsilon_i} I$, $\epsilon_i > 0$, then whenever $\epsilon_i < \frac{1}{\delta_i \| B_i \|_2^2}$ holds for all $i \in \{1,2\}$, $\| \Phi_i(X) - \Phi_i(Y) \|_2 \leq \frac{1}{(1-\nu_i)^2} \| X - Y \|_2$ where $\nu_i \triangleq \epsilon_i \delta_i \| B_i \|_2^2 < 1$.

***Proof* :** Under the given assumptions, it can be shown that :

$$\begin{aligned} & \| \Phi_i(X) - \Phi_i(Y) \|_2 \\ &= \left\| \left[I + \epsilon_i X B_i B_i^T \right]^{-1} \left[X - Y \right] \left[I + \epsilon_i B_i B_i^T Y \right]^{-1} \right\|_2 \\ &\leq \left\| \left[I + \epsilon_i X B_i B_i^T \right]^{-1} \right\|_2 \cdot \left\| \left[I + B_i B_i^T Y \right]^{-1} \right\|_2 \cdot \| X - Y \|_2 . \end{aligned} \quad \text{(B-F.3.2.1)}$$

Since $\nu_i < 1$ for all $i \in \{1,2\}$,

$$\begin{aligned} \left\| \left[I + \epsilon_i X B_i B_i^T \right]^{-1} \right\|_2 &= \frac{1}{\underline{\sigma}(I + \epsilon_i X B_i B_i^T)} \\ &\leq \frac{1}{1 - \overline{\sigma}(\epsilon_i X B_i B_i^T)} \\ &\leq \frac{1}{1 - \overline{\sigma}(\epsilon_i \delta_i B_i B_i^T)} \\ &\leq \frac{1}{1 - \nu_i} \end{aligned} \quad \text{(B-F.3.2.2)}$$

where $\overline{\sigma}(\cdot) = \| \cdot \|_2$ is the largest singular value of $(\cdot)$. Similarly,

$$\left\| \left[I + \epsilon_i B_i B_i^T Y \right]^{-1} \right\|_2 \leq \frac{1}{1 - \nu_i} . \quad \text{(B-F.3.2.3)}$$

Hence, in view of (B-F.3.2.2)-(B-F.3.2.3), (B-F.3.2.1) becomes

$$\| \Phi_i(X) - \Phi_i(Y) \|_2 \leq \frac{1}{(1-\nu_i)^2} \| X - Y \|_2 .$$

□

***Fact* 3.3 :** Given any $(X, Y) \in \Delta \triangleq B_{\bar{\delta}_1} \times B_{\bar{\delta}_2}$ and $R_i \triangleq \frac{1}{\epsilon_i} I$, $\epsilon_i > 0$, then whenever $0 < \epsilon_i < \bar{\epsilon}_i(\bar{\delta}_i) \triangleq \frac{1}{\bar{\delta}_i \| B_i \|_2^2}$ holds for all $i \in \{1,2\}$, it follows that

$$\rho_1(X, Y) = \| A_1(X, Y) \|_2 \leq \xi \frac{1 + \bar{\nu}_2}{1 - \bar{\nu}_1 \bar{\nu}_2} \triangleq \xi \bar{\kappa}_1 , \quad \bar{\kappa}_1 \geq 1$$

$$\rho_2(X, Y) = \| A_2(X, Y) \|_2 \leq \xi \frac{1 + \bar{\nu}_1}{1 - \bar{\nu}_1 \bar{\nu}_2} \triangleq \xi \bar{\kappa}_2 , \quad \bar{\kappa}_2 \geq 1$$

where $\bar{\nu}_i \triangleq \epsilon_i \bar{\delta}_i \| B_i \|_2^2$ and $\xi \triangleq \| A \|_2$.

***Proof* :** By definition,

$$\| A_1(X, Y) \|_2 = \| A - B_2 F_2(X, Y) \|_2 \leq \| A \|_2 + \| B_2 F_2(X, Y) \|_2$$

$$\| A_2(X, Y) \|_2 = \| A - B_1 F_1(X, Y) \|_2 \leq \| A \|_2 + \| B_1 F_1(X, Y) \|_2 .$$

Lemma 5 of the Appendix states that

$$\| B_1 F_1(X, Y) \|_2 \leq \xi \frac{\bar{\nu}_1 (1 + \bar{\nu}_2)}{1 - \bar{\nu}_1 \bar{\nu}_2} \text{ and } \| B_2 F_2(X, Y) \|_2 \leq \xi \frac{\bar{\nu}_2 (1 + \bar{\nu}_1)}{1 - \bar{\nu}_1 \bar{\nu}_2}$$

where $\xi \triangleq \| A \|_2$. Hence,

$$\| A_1(X, Y) \|_2 \leq \xi + \xi \frac{\bar{\nu}_2 (1 + \bar{\nu}_1)}{1 - \bar{\nu}_1 \bar{\nu}_2}$$

$$= \xi \left| 1 + \frac{\bar{\nu}_2 (1 + \bar{\nu}_1)}{1 - \bar{\nu}_1 \bar{\nu}_2} \right| = \xi \left| \frac{(1 - \bar{\nu}_1 \bar{\nu}_2) + \bar{\nu}_2 + \bar{\nu}_1 \bar{\nu}_2}{1 - \bar{\nu}_1 \bar{\nu}_2} \right|$$

$$= \xi \left| \frac{1 + \bar{\nu}_2}{1 - \bar{\nu}_1 \bar{\nu}_2} \right| .$$

Defining $\bar{\kappa}_1 \triangleq \frac{1 + \bar{\nu}_2}{1 - \bar{\nu}_1 \bar{\nu}_2}$, then $\| A_1(X, Y) \|_2 \leq \bar{\kappa}_1 \xi$. Similarly,

$$\| A_2(X, Y) \|_2 \leq \xi \left| \frac{1 + \bar{\nu}_1}{1 - \bar{\nu}_1 \bar{\nu}_2} \right| \triangleq \bar{\kappa}_2 \xi .$$

Since $0 < \nu_i < 1$, then $\inf_{\bar{\nu}_1, \bar{\nu}_2} \{ \bar{\kappa}_i \} = 1$ and occurs at $\bar{\nu}_1 = \bar{\nu}_2 = 0$. Thus, $\bar{\kappa}_1, \bar{\kappa}_2 \geq 1$.

□

Some Important Facts Which Summarize the Lemmas

In all cases, $i \in \{ 1,2 \}$.

1) $$\left\| \left[\Gamma_i(X)\right]^{-1} \right\| \leq \epsilon_i \text{ for all } X \text{ and all } \epsilon_i > 0 . \qquad \text{(B-2.4)}$$

2) $$\left\| \left[\Gamma_i(X)\right]^{-1} - \left[\Gamma_i(Y)\right]^{-1} \right\| \leq (\epsilon_i)^2 \| B_i \|^2 \cdot \| X - Y \| \qquad \text{(LEMMA1)}$$
for all X and Y and all $\epsilon_i > 0$.

3) $$\| \Psi_i(X) \| \leq \epsilon_i \| B_i \| \cdot \| X \| \text{ for all } X \text{ and all } \epsilon_i > 0 . \qquad \text{(B-3.5)}$$

4) $$\| \Psi_i(X) - \Psi_i(Y) \| \leq \epsilon_i \| B_i \| \cdot \left| 1 + \frac{\epsilon_i}{2} \| B_i \|^2 \cdot \| X + Y \| \right| \cdot \| X - Y \| \qquad \text{(LEMMA2)}$$
for all X, Y and all $\epsilon_i > 0$. Moreover, if X, $Y \in B_{\delta_i}$ then

$$\| \Psi_i(X) - \Psi_i(Y) \| \leq \epsilon_i \| B_i \| \cdot \left| 1 + \epsilon_i \delta_i \| B_i \|^2 \right| \cdot \| X - Y \|$$
$$\stackrel{\Delta}{=} \epsilon_i \| B_i \| \cdot (1 + \nu_i) \cdot \| X - Y \| .$$

5) $$\left\| \left[\Xi_i(X_1, X_2)\right]^{-1} \right\| \leq \frac{1}{1 - \lambda_1(X_1)\lambda_2(X_2)} \leq \frac{1}{1 - \nu_1 \nu_2} \qquad \text{(B-4.5)}$$
whenever $\epsilon_i < \dfrac{1}{\delta_i \| B_i \|^2}$ where $X_1 \in B_{\delta_1}$.

APPENDIX C

L-A-S CONTRACTION MAPPING PROGRAM RUN

This appendix contains an L-A-S program to check for a contraction mapping and a sample run of the program. A summary and discussion of the data generated here may be found in subsection 3.4.3.

```
>rpf.contra

>pro
  1 ; Linear Quadratic Nash Game
  2 (rdf)=a,b1,b2,c1,c2
  3 a,b1,b2,c1,c2(out,e)=
  4 a,b1,b2,c1,c2(game)=
  5 (rdf)=s1,r1,s2,r2
  6 r1,r2,s1,s2(out)=
  7 c1(t),s1(*),c1(*)=q1
  8 c2(t),s2(*),c2(*)=q2
  9 q1,r1,q2,r2(lq)=
 10 (rdf)=z1,z2
 11 q1,q2(mcp)=k1,k2
 12 ;
 13 1(dsc)=one
 14 "Enter the total number of stages in this game."
 15 (inp)=ii
 16 ; Main Loop
 17 (stop)=
 18 a:k1,k2(out,e)=
 19 k1,k2(lqng)=k1n,k2n
 20 k1n,k2n(out,e)=
 21 ; Contraction Mapping Constant
 22 k1,z1(-)=zz1
 23 k2,z2(-)=zz2
 24 k1n,k1(-)=tt1
 25 k2n,k2(-)=tt2
 26 zz1(nrm2)=xz1
 27 zz2(nrm2)=xz2
 28 tt1(nrm2)=xt1
 29 tt2(nrm2)=xt2
 30 xt1,xt2(+)=xt
 31 xz1,xz2(+)=xz
 32 xz(inv)=xzi
 33 xt,xzi(*)=alf
 34 xz1,xz2,alf(out,e)=
 35 k1,k2(mcp)=z1,z2
 36 k1n,k2n(mcp)=k1,k2
 37 ii,one(-)=ii
 38 ii(if)=a
```

```
>con

>: Linear Quadratic Nash Game

>(rdf)=a,b1,b2,c1,c2

Enter name of the Data File (DF) for matrix a    >contr1
Opening file named : contr1.DF
Reading array named : a
Reading array named : b1
Reading array named : b2
Reading array named : c1
Reading array named : c2

>a,b1,b2,c1,c2(out,e)=

      a
 4.75537e-01   4.58790e-02  -1.13295e-04
 4.58790e-02   3.44463e-01  -7.17458e-05
-1.13295e-04  -7.17458e-05   2.50000e-01

      b1
 9.87000e+02
 1.23000e+00
-1.01000e-03

      b2
 1.37000e+00
-1.00000e-03
 1.00000e-05

      c1
 1.00000e+00   0.00000e+00   0.00000e+00
 0.00000e+00   1.00000e+00   0.00000e+00
 0.00000e+00   0.00000e+00   1.00000e+00

      c2
 1.00000e+00   0.00000e+00   0.00000e+00
 0.00000e+00   1.00000e+00   0.00000e+00
 0.00000e+00   0.00000e+00   1.00000e+00

>a,b1,b2,c1,c2(game)=

>(rdf)=s1,r1,s2,r2

Enter name of the Data File (DF) for matrix s1    >contr2
Opening file named : contr2.DF
Reading array named : s1
Reading array named : r1
Reading array named : s2
Reading array named : r2

>r1,r2,s1,s2(out)=
```

```
      r1
11.000

      r2
11.000

      s1
 1.000  0.     0.
 0.     1.000  0.
 0.     0.     1.000

      s2
 1.000  0.     0.
 0.     1.000  0.
 0.     0.     1.000

 >c1(t),s1(*),c1(*)=q1
  #1,s1(*),c1(*)=q1
  #2,c1(*)=q1

 >c2(t),s2(*),c2(*)=q2
  #1,s2(*),c2(*)=q2
  #2,c2(*)=q2

 >q1,r1,q2,r2(lq)=

 >(rdf)=z1,z2

Enter name of the Data File (DF) for matrix z1   >contr3
Opening file named : contr3.DF
Reading array named : z1
Reading array named : z2

 >q1,q2(mcp)=k1,k2

 >:

 >1(dsc)=one

 >"Enter the total number of stages in this game."

Enter the total number of stages in this game.

 >(inp)=ii

*** Matrix ii ***
Enter the dimensions of this matrix.  >1,1

Enter the scalar : ii    >15

 >: Main Loop

 >(stop)=
```

```
>nli

>con

     k1
 1.00000e+00   0.00000e+00   0.00000e+00
 0.00000e+00   1.00000e+00   0.00000e+00
 0.00000e+00   0.00000e+00   1.00000e+00

     k2
 1.00000e+00   0.00000e+00   0.00000e+00
 0.00000e+00   1.00000e+00   0.00000e+00
 0.00000e+00   0.00000e+00   1.00000e+00

     k1n
 1.00205e+00   1.55971e-02  -3.14454e-05
 1.55971e-02   1.11861e+00  -4.25852e-05
-3.14454e-05  -4.25852e-05   1.06250e+00

     k2n
 1.00205e+00   1.55969e-02  -3.14449e-05
 1.55969e-02   1.11862e+00  -4.25854e-05
-3.14449e-05  -4.25854e-05   1.06250e+00

     xz1
 1.00000e+00

     xz2
 1.00000e+00

     alf
 1.20666e-01

     k1
 1.00205e+00   1.55971e-02  -3.14454e-05
 1.55971e-02   1.11861e+00  -4.25852e-05
-3.14454e-05  -4.25852e-05   1.06250e+00

     k2
 1.00205e+00   1.55969e-02  -3.14449e-05
 1.55969e-02   1.11862e+00  -4.25854e-05
-3.14449e-05  -4.25854e-05   1.06250e+00

     k1n
 1.00230e+00   1.74427e-02  -3.40677e-05
 1.74427e-02   1.13265e+00  -5.02427e-05
-3.40677e-05  -5.02427e-05   1.06641e+00

     k2n
 1.00229e+00   1.74424e-02  -3.40671e-05
 1.74424e-02   1.13265e+00  -5.02429e-05
-3.40671e-05  -5.02429e-05   1.06641e+00
```

xz1
1.20666e-01

xz2
1.20666e-01

alf
1.18329e-01

k1
1.00230e+00 1.74427e-02 -3.40677e-05
1.74427e-02 1.13265e+00 -5.02427e-05
-3.40677e-05 -5.02427e-05 1.06641e+00

k2
1.00229e+00 1.74424e-02 -3.40671e-05
1.74424e-02 1.13265e+00 -5.02429e-05
-3.40671e-05 -5.02429e-05 1.06641e+00

k1n
1.00232e+00 1.76606e-02 -3.43086e-05
1.76606e-02 1.13431e+00 -5.13070e-05
-3.43086e-05 -5.13070e-05 1.06665e+00

k2n
1.00232e+00 1.76603e-02 -3.43080e-05
1.76603e-02 1.13431e+00 -5.13073e-05
-3.43080e-05 -5.13073e-05 1.06665e+00

xz1
1.42782e-02

xz2
1.42782e-02

alf
1.18053e-01

k1
1.00232e+00 1.76606e-02 -3.43086e-05
1.76606e-02 1.13431e+00 -5.13070e-05
-3.43086e-05 -5.13070e-05 1.06665e+00

k2
1.00232e+00 1.76603e-02 -3.43080e-05
1.76603e-02 1.13431e+00 -5.13073e-05
-3.43080e-05 -5.13073e-05 1.06665e+00

k1n
1.00233e+00 1.76863e-02 -3.43328e-05
1.76863e-02 1.13450e+00 -5.14426e-05
-3.43328e-05 -5.14426e-05 1.06667e+00

k2n
1.00233e+00 1.76860e-02 -3.43321e-05
1.76860e-02 1.13450e+00 -5.14429e-05
-3.43321e-05 -5.14429e-05 1.06667e+00

xz1
1.68558e-03

xz2
1.68558e-03

alf
1.18020e-01

k1
1.00233e+00 1.76863e-02 -3.43328e-05
1.76863e-02 1.13450e+00 -5.14426e-05
-3.43328e-05 -5.14426e-05 1.06667e+00

k2
1.00233e+00 1.76860e-02 -3.43321e-05
1.76860e-02 1.13450e+00 -5.14429e-05
-3.43321e-05 -5.14429e-05 1.06667e+00

k1n
1.00233e+00 1.76893e-02 -3.43353e-05
1.76893e-02 1.13453e+00 -5.14592e-05
-3.43353e-05 -5.14592e-05 1.06667e+00

k2n
1.00233e+00 1.76890e-02 -3.43347e-05
1.76890e-02 1.13453e+00 -5.14595e-05
-3.43347e-05 -5.14595e-05 1.06667e+00

xz1
1.98932e-04

xz2
1.98933e-04

alf
1.18016e-01

k1
1.00233e+00 1.76893e-02 -3.43353e-05
1.76893e-02 1.13453e+00 -5.14592e-05
-3.43353e-05 -5.14592e-05 1.06667e+00

k2
1.00233e+00 1.76890e-02 -3.43347e-05
1.76890e-02 1.13453e+00 -5.14595e-05
-3.43347e-05 -5.14595e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43356e-05
1.76897e-02 1.13453e+00 -5.14613e-05
-3.43356e-05 -5.14613e-05 1.06667e+00

k2n
1.00233e+00 1.76894e-02 -3.43350e-05
1.76894e-02 1.13453e+00 -5.14615e-05
-3.43350e-05 -5.14615e-05 1.06667e+00

xz1
2.34772e-05

xz2
2.34773e-05

alf
1.18016e-01

k1
1.00233e+00 1.76897e-02 -3.43356e-05
1.76897e-02 1.13453e+00 -5.14613e-05
-3.43356e-05 -5.14613e-05 1.06667e+00

k2
1.00233e+00 1.76894e-02 -3.43350e-05
1.76894e-02 1.13453e+00 -5.14615e-05
-3.43350e-05 -5.14615e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2n
1.00233e+00 1.76894e-02 -3.43350e-05
1.76894e-02 1.13453e+00 -5.14618e-05
-3.43350e-05 -5.14618e-05 1.06667e+00

xz1
2.77068e-06

xz2
2.77069e-06

alf
1.18016e-01

k1
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2
1.00233e+00 1.76894e-02 -3.43350e-05
1.76894e-02 1.13453e+00 -5.14618e-05
-3.43350e-05 -5.14618e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2n
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

xz1
3.26983e-07

xz2
3.26986e-07

alf
1.18016e-01

k1
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2n
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

xz1
3.85891e-08

xz2
3.85894e-08

alf
1.18016e-01

k1
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2n
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

xz1
4.55412e-09

xz2
4.55416e-09

alf
1.18016e-01

k1
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2n
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

xz1
5.37457e-10

xz2

5.37463e-10

alf
1.18016e-01

k1
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2n
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

xz1
6.34284e-11

xz2
6.34291e-11

alf
1.18016e-01

k1
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2n
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

xz1
7.48554e-12

xz2
7.48565e-12

alf
1.18014e-01

k1
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2n
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

xz1
8.83404e-13

xz2
8.83404e-13

alf
1.18020e-01

k1
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

k2
1.00233e+00 1.76895e-02 -3.43351e-05
1.76895e-02 1.13453e+00 -5.14618e-05
-3.43351e-05 -5.14618e-05 1.06667e+00

k1n
1.00233e+00 1.76897e-02 -3.43357e-05
1.76897e-02 1.13453e+00 -5.14615e-05
-3.43357e-05 -5.14615e-05 1.06667e+00

```
      k2n
 1.00233e+00   1.76895e-02  -3.43351e-05
 1.76895e-02   1.13453e+00  -5.14618e-05
-3.43351e-05  -5.14618e-05   1.06667e+00

      xz1
 1.04246e-13

      xz2
 1.04273e-13

      alf
 1.18080e-01

>end
```

REFERENCES

[1] M. Athans and P. L. Falb, *Optimal Control: An Introduction to the Theory and Its Applications*. New York : McGraw-Hill, 1966.

[2] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*. New York : Wiley-Interscience, 1972.

[3] A. E. Bryson, Jr. and Y. C. Ho, *Applied Optimal Control*. New York : Hemisphere Publishing Company, 1975.

[4] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions ASME Journal Basic Engineering*, vol. 82, pp. 34-35, March 1960.

[5] D. P. Bertsekas, *Dynamic Programming and Stochastic Control*. New York : Academic Press, 1976.

[6] B. C. Kuo, *Digital Control Systems*. New York : Holt, Rinehart and Winston, 1980.

[7] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. New York : Academic Press, 1982.

[8] D. L. Lukes, "Equilibrium Feedback Control in Linear Games with Quadratic Costs," *SIAM Journal on Control*, vol. 9, no. 2, pp. 234-252, 1971.

[9] E. F.Mageirou, "Values and Strategies for Infinite-Time Linear Quadratic Games," *IEEE Transactions on Automatic Control*, vol. AC-21, no. 4, pp. 547-550, 1976.

[10] E. F. Mageirou, "Iterative Techniques for Riccati Game Equations," *Journal of Optimization Theory and Applications*, vol. 22, no. 1, pp. 51-61, 1977.

[11] G. P. Papavassilopoulos, J. V. Medanic, and J. B. Cruz, Jr., "On the Existence of Nash Strategies and Solutions to Coupled Riccati Equations in Linear Quadratic Games," *Journal of Optimization Theory and Applications*, vol. 28, no. 1, pp. 49-76, 1979.

[12] G. P. Papavassilopoulos, "On the LQ, Closed-loop, No Memory Nash Game," *Journal of Optimization Theory and Applications*, vol. 42, no. 4, pp. 551-560, April 1984.

[13] G. P. Papavassilopoulos, "Leader-Follower and Nash Strategies with State Information," Ph.D. dissertation, CSL Report R-852 (DC-29), Coordinated Science Laboratory, University of Illinois, Urbana, July 1979.

[14] Abstracts from the *IEEE Control Systems Society Symposium on Computer-Aided Control System Design*, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 28-30, 1983.

[15] *Proceedings of the 2nd IEEE Control Systems Society Symposium on Computer-Aided Control System Design*, Santa Barbara, California, March 13-15, 1985.

[16] *Proceedings of the 3rd IFAC Symposium on Computer Aided Design in Control and Engineering Systems*, Copenhagen, Denmark, July 31-August 2, 1985.

[17] G. W. Stewart, *Introduction to Matrix Computations*. New York : Academic Press, 1973.

[18] V. C. Klema and A. J. Laub, "The Singular Value Decomposition: Its Computation and Some Applications," *IEEE Transactions on Automatic Control*, vol. AC-25, no. 2, pp. 164-176, April 1980.

[19] A. J. Laub, "Schur Techniques in Invariant Imbedding Methods for Solving Two-Point Boundary Value Problems," *Proceedings of the 21st IEEE Conference on Decision and Control*, Orlando, Florida, pp. 56-61, December 1982.

[20] J. D. Cobb, "Descriptor Variable and Generalized Singularly Perturbed Systems: A Geometric Approach," Ph.D. dissertation, CSL Report R-882 (DC-38), Coordinated Science Laboratory, Urbana, Illinois, 1980.

[21] A. J. Laub, "Numerical Linear Algebra Aspects of Control Design Computations," *IEEE Transactions on Automatic Control*, vol. AC-30, no. 2, pp. 97-108, February 1985.

[22] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines - EISPACK Guide, 2nd edition*. Lecture Notes in Computer Science, vol. 6, New York : Springer-Verlag, 1976.

[23] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler, *Matrix Eigensystem Routines - EISPACK Guide Extension*. Lecture Notes in Computer Science, vol. 51, New York : Springer-Verlag, 1977.

[24] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK User's Guide*. Philadelphia : SIAM, 1979.

[25] H. Katzan, Jr., *FORTRAN 77*. New York : Van Nostrand Reinhold Company, 1978.

[26] W. F. Arnold and A. J. Laub, "A Software Package for the Solution of Generalized Algebraic Riccati Equations," *Proceedings of the 22nd IEEE Conference on Decision and Control*, San Antonio, TX, December 1983.

[27] M. Jamshidi and C. J. Herget, Eds., *Computer-Aided Control Systems Engineering*. New York : North-Holland, 1985.

[28] H. Elmqvist, "SIMNON: An Interactive Simulation Program for Nonlinear Systems - User's Manual," Report 7502, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, April 1975.

[29] S. P. Bingulac and M. A. Faries, "L-A-S (Linear Algebra and Systems) Language and Its Use in Control Education and Research," *Computing*, vol. 23, Springer-Verlag, pp. 1-23, 1979.

[30] G. R. McArthur, Ed., *The SCD Graphics Utilities*, National Center for Atmospheric Research (NCAR) Technical Note NCAR-TN/166+IA, Boulder, Colorado, February 1981.

[31] S. P. Bingulac and N. Gluhajic, "Computer Aided Design of Control Systems on Mini Computers Using the L-A-S Language," *Proceedings of 1982 IFAC Symposium on Computer Aided Design of Multivariable Technological Systems*, Purdue University, Indiana, September 15-17, 1982.

[32] S. P. Bingulac, "Recent Modifications in the L-A-S (Linear Algebra and Systems) Language and Its Use in CAD of Control Systems," *Proceedings of 1983 Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois, pp. 393-402, October 1983.

[33] S. P. Bingulac and P. J. West, "Computer-Aided Design of Control Systems: An Example of Working Software," *Proceedings of 21st Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois, pp. 423-424, October 1983.

[34] S. P. Bingulac, J. H. Chow, S. H. Javid, and H. R. Dowse, "User's Manual for Linear Algebra and Systems (L-A-S) Language," Report No. 83-EUE-205, (Internal Report), Electric Utility Systems Engineering Department, General Electric Co., Schenectady, NY 12345, November 1983.

[35] P. J. West, S. P. Bingulac, and W. R. Perkins, "Recent Advances in the L-A-S Language Used for CACSD," *Proceedings of the 2nd IEEE Control Systems Society Symposium on Computer-Aided Control System Design*, Santa Barbara, CA, March 13-15, 1985.

[36] S. P. Bingulac, P. J. West, and W. R. Perkins, "Recent Advances in the L-A-S Software Used in CAD of Control Systems," *Proceedings of the 3rd IFAC Symposium on Computer Aided Design in Control and Engineering Systems*, Copenhagen, Denmark, July 31-August 2, 1985.

[37] P. V. Kokotovic, R. E. O'Malley, Jr. and P. Sannuti, "Singular Perturbations and Order Reduction in Control Theory - An Overview," *Automatica*, vol. 12, pp. 123-132, 1976.

[38] H. K. Khalil, Ed., "Singular Perturbation Methods and Multimodel Control," Final Report for DOE Contract DE-AC01-80RA50425, Michigan State University, East Lansing, Michigan, December 1983.

[39] P. J. West and W. R. Perkins, "Nash Strategies for Discrete-Time Linear Systems with Multirate Controllers," *Proceedings of the Fifth American Control Conference*, Seattle, Washington, June 18-20, 1986.

[40] T. Pappas, A. J. Laub, and N. R. Sandell, Jr., "On the Numerical Solution of the Discrete-Time Algebraic Riccati Equation," *IEEE Transactions on Automatic Control*, vol. AC-25, no. 4, pp. 631-641, August 1980.

[41] W. F. Arnold, "On the Numerical Solution of Algebraic Riccati Equations," Ph.D. dissertation, Department of Electrical Engineering and Systems, University of Southern California, Los Angeles, California, December 1983.

[42] K. H. Lee, "Generalized Eigenproblem Structures and Solution Methods for Riccati Equations," Ph.D. dissertation, Department of Electrical Engineering and Systems, University of Southern California, Los Angeles, California, January 1983.

[43] D. J. Bender, "Descriptor Systems and Geometric Control Theory," Ph.D. dissertation, Department of Electrical and Computer Engineering, University of California, Santa Barbara, California, 1985.

[44] D. J. Bender and A. J. Laub, "The Linear-Quadratic Optimal Regulator for Descriptor Systems : Discrete-Time Case," submitted for publication.

[45] V. R. Saksena, J. B. Cruz, Jr., W. R. Perkins, and T. Basar, "Information Induced Multimodel Solutions in Multiple Decisionmaker Problems," *IEEE Transactions on Circuits and Systems, Joint Special Issue on Large Scale Systems*, vol. CAS-30, no. 6, pp. 403-415, June 1983.

[46] M. Athans, "The matrix minimum principle," *Information Control*, vol. 11, pp. 592-606, November 1967.

[47] C. C. Paige, "Properties of Numerical Algorithms Related to Computing Controllability," *IEEE Transactions on Automatic Control*, vol. AC-26, no. 1, pp. 130-138, February 1981.

[48] S. W. Chan, G. C. Goodwin, and K. S. Sin, "Convergence Properties of the Riccati Difference Equation in Optimal Filtering of Nonstabilizable Systems," *IEEE Transactions on Automatic Control*, vol. AC-29, no. 2, pp. 110-118, February 1984.

VITA

Phillip James West was born in Seattle, Washington, on August 28, 1958. He received the Bachelor of Science degree in Computer Engineering from the University of Illinois at Urbana-Champaign in 1980. The Master of Science degree in Electrical Engineering was received from the same institution in 1982.

Since August 1982, he has been a graduate student at the University of Illinois where he has worked as a research assistant in the Coordinated Science Laboratory. His research interests include numerical algorithms, computer-aided design, and game theory. In December 1985, he joined the BDM Corporation in Hawthorne, California.

Phillip J. West is a member of the Institute of Electrical and Electronics Engineers.

END

10-86

DTIC